

An Evaluation Paradigm for Spoken Dialog Systems based on Crowdsourcing and Collaborative Filtering

YANG, Zhaojun

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

The Chinese University of Hong Kong

September 2011

Thesis/Assessment Committee

Professor K. F. Wong(Chair)

Professor Helen Meng(Thesis Supervisor)

Professor C. H. Cheng(Committee Member)

Professor Maxine Eskenazi(External Examiner)

Abstract

This thesis proposes an evaluation paradigm for spoken dialog systems (SDSs), in order to facilitate SDS development. It is consisting of two components, namely collecting user judgments on an SDS and developing an evaluation model based on the collected judgments. First, a crowdsourcing method is proposed to collect user judgments on the SDS's performance through Amazon Mechanical Turk (MTurk). Two types of tasks are designed – one targets at fast rating a large number of dialogs regarding some dimensions of the SDS's performance and the other aims to assess the reliability of Workers on MTurk. Second, a collaborative filtering (CF) model is developed for automatic SDS evaluation based on the crowdsourced judgments. It predicts user satisfaction about the overall quality of an SDS by exploiting the information from the neighbors of an unrated dialog. The proposed methodology is tested on the dialog corpus from the Let's Go! system which is developed by the Dialog Research Center at CMU. Experimental results show that reliable ratings of a number of dialogs can be obtained from MTurk efficiently and cost-effectively with a high level of agreement between experts and MTurk Workers, and that CF models can significantly improve the prediction performance of previous methods. This methodology is also applied to evaluate the performance of our bus information dialog system which is built based on the partially observable Markov decision process (POMDP).

摘要

對話系統的評價能有效地促進系統的設計和發展。在這篇論文中，我們提出了一種系統評價模型。該模型分為兩個部份，即收集用戶對系統的評價以及基於收集的評價開發出自動評價系統性能的方法。首先，針對傳統用戶實驗的昂貴耗時的缺陷，我們提出一種群眾外包的方法通過 Amazon Mechanical Turk (MTurk) 來收集用戶對系統性能的評價。我們設計了兩種任務，分別用於快速收集用戶的系統評價和驗證 MTurk 工人的可靠性。其次，我們提出一種基於協同過濾法的評價方法來自動預測用戶對系統性能的滿意度。該模型利用未評價對話的相似對話的信息來對此未評價對話的用戶滿意度做出預測。該系統評價模型在來自 Let's Go! 對話系統的對話庫上進行測試。Let's Go! 系統是由卡耐基梅隆大學的對話研究中心開發的。實驗結果表明在專家和 MTurk 工人之間很高的一致性下，群眾外包法能有效低廉地收集大量對話的用戶評價。相對傳統的評價模型 PARADISE，協同過濾評價方法能有效的提高預測用戶滿意度的準確率。同時，該評價模型也被用於評價我們的基於部份可觀察馬爾科夫決策過程所建立的對話系統。

Acknowledgement

First, I would like to thank my supervisor, Professor Helen Meng, who leads me to the research road, provides helpful guidance and precious advice to me, and strongly supports me for pursuing my goal; and from whom I learnt the industrious working spirits and the ability to think, write and communicate.

I would like to thank other professors in our collaborative project, Professor Gina-Anne Levow and Professor Irwin King. Thanks for their enlightening discussions and a great deal of insightful suggestions. Thanks also to Baichuan and Yi at the Department of Computer Science and Engineering for the effective discussions and the wonderful cooperations during the 2010 SDC.

Many thanks also go to our team members in HCCL, in particular Dr. Lo Weiwu, Xiaojun, Kunge, Yanyi, Jackie, Mingxing, and Shuang. Thanks for your favors, and sharing your happiness during my time. You make HCCL a very happy and excellent place to do research.

Thanks also to my dear friends, Xiaohai, Yiwa, Tuzi, Xiong, Mengwa, Xuewa, Haizai, and Yu, for your soul companies, comforts, and friendship which has been and always will be a priceless treasure in my life.

Finally, I would like to express deep appreciation to my parents and GBQ, for your enduring love; for your encouragement when I feel frustrated; and for you unfailing support all the way.

Contents

1	Introduction	1
1.1	SDS Architecture	1
1.2	Dialog Model	3
1.3	SDS Evaluation	4
1.4	Thesis Outline	7
2	Previous Work	9
2.1	Approaches to Dialog Modeling	9
2.1.1	Handcrafted Dialog Modeling	9
2.1.2	Statistical Dialog Modeling	12
2.2	Evaluation Metrics	16
2.2.1	Subjective User Judgments	17
2.2.2	Interaction Metrics	18
2.3	The PARADISE Framework	19
2.4	Chapter Summary	22
3	Implementation of a Dialog System based on POMDP	23
3.1	Partially Observable Markov Decision Processes (POMDPs) . . .	24
3.1.1	Formal Definition	24
3.1.2	Value Iteration	26
3.1.3	Point-based Value Iteration	27
3.1.4	A Toy Example of POMDP: The NaiveBusInfo System . . .	27

3.2	The SDS-POMDP Model	31
3.3	Composite Summary Point-based Value Iteration (CSPBVI)	33
3.4	Application of SDS-POMDP Model: The BusInfo System	35
3.4.1	System Description	35
3.4.2	Demonstration Description	39
3.5	Chapter Summary	42
4	Collecting User Judgments on Spoken Dialogs with Crowdsourcing	46
4.1	Dialog Corpus and Automatic Dialog Classification	47
4.2	User Judgments Collection with Crowdsourcing	50
4.2.1	HITs on Dialog Evaluation	51
4.2.2	HITs on Inter-rater Agreement	53
4.2.3	Approval of Ratings	54
4.3	Collected Results and Analysis	55
4.3.1	Approval Rates and Comments from Mturk Workers	55
4.3.2	Consistency between Automatic Dialog Classification and Manual Ratings	57
4.3.3	Inter-rater Agreement Among Workers	60
4.4	Comparing Experts to Non-experts	64
4.4.1	Inter-rater Agreement on the Let's Go! System	65
4.4.2	Consistency Between Expert and Non-expert Annotations on SDC Systems	66
4.5	Chapter Summary	68
5	Collaborative Filtering for Performance Prediction	70

5.1	Item-Based Collaborative Filtering	71
5.2	CF Model for User Satisfaction Prediction	72
5.2.1	ICFM for User Satisfaction Prediction	72
5.2.2	Extended ICFM for User Satisfaction Prediction	73
5.3	Extraction of Interaction Features	74
5.4	Experimental Results and Analysis	76
5.4.1	Prediction of User Satisfaction	76
5.4.2	Analysis of Prediction Results	79
5.5	Verifying the Generalibility of CF Model	81
5.6	Evaluation of The BusInfo System	86
5.7	Chapter Summary	87
6	Conclusions and Future Work	89
6.1	Thesis Summary	89
6.2	Future Work	90
	Bibliography	92

List of Figures

1.1	A typical spoken dialog turn between the system and the user . . .	2
2.1	A part of the tree of agents in the RavenClaw architecture [1]. . .	13
2.2	The PARADISE structure of objectives for spoken dialog performance . . .	20
3.1	A POMDP example in the NaiveBusInfo system.	28
3.2	The process of belief updating based on the observation and the previous action of the system.	29
3.3	Illustration of the optimal policy in a POMDP framework. . . .	30
3.4	The optimal policy of the NaiveBusInfo system.	30
3.5	The factor graph of SDS-POMDP	32
3.6	The architecture of BusInfo system	39
3.7	Screen shot of the BusInfo system showing the beginning of the dialog. It consists of four parts, <i>i.e.</i> , interaction part, dialog status part, dialog history part, and covered information part. In the beginning of the dialog, the initial belief over the user goals for each slot is uniform.	40
4.1	Samples of the Let's Go! dialogs in the categories of automatic classification.	49
4.2	Distribution of the five dialog categories based on the automatic, heuristics-based classification.	50

4.3	Approval rates for each dialog category, i.e., TaskComplete, TaskIncomplete and OutofScope. OutofScope has the highest approval rate.	56
4.4	Percentages of ambiguous dialogs over the total approved ones in each category of TaskComplete, TaskIncomplete and OutofScope. A higher percentage of ambiguous dialogs are approved for the OutOfScope category.	58
4.5	The normalized mean scores of Q2 and Q5 for approved ratings in each category. A higher score maps to a higher level of task success.	61
4.6	Mean values of weighted Kappa of five questions for 3 groups. The Kappa values are stable across different raters and different dialogs. Q2 and Q5 which can gain “official” ratings from reliable raters have high levels of inter-rater agreement.	64
5.1	R^2 for user satisfaction prediction, in relation to the number of clusters M for the three prediction models. EICFM shows distinct improvement and is less sensitive to M	77
5.2	Average ratings of user satisfaction for dialogs over different #System Turns. The solid line with circle markers is for true ratings, and the other lines are for the predictions.	78
5.3	The probability density plots of #System Questions, #System Turns, AveRecogScore and #Barge In for dialogs rated high (A), medium (B) and low (C). The plots of other features are similar to that of #Barge In. The densities of A , B and C on AveRecogScore, #System Turns and #System Questions are different, which implies the three features have relatively larger impact on user satisfaction.	80

- 5.4 Evaluation results for the SDSs from CF and MTurk. The mean
scores in terms of Su of the four SDC systems predicted by the
CF model are quite similar to those rated by MTurk Workers . . 84
- 5.5 Normalized mean values with standard deviations of dialog features
of Let's Go! and BusInfo systems. 87

List of Tables

3.1	Description of the components for the master and summary POMDPs. The <i>predicate</i> refers to the illocutionary force of the action. \hat{x} stands for the component in the summary space. n means the slot value has not been stated, u means the slot value has not been confirmed, and c means the slot value has been confirmed.	34
3.2	Summarizations of the slots information in the BusInfo system .	36
3.3	Summarizations of the components in the BusInfo system	36
3.4	Summarizations of user action model in the BusInfo system . . .	37
3.5	Summarizations of reward model in the BusInfo system	38
3.6	An example of the BusInfo dialogs.	42
3.7	An example dialog with the BusInfo system. In each turn, the upper part shows the user-system interaction, while the lower part shows the dialog status, including the belief of the most likely user goal in each slot, the proposed actions, and the most likely user goals.	45
4.1	An example of the Let's Go! dialogs.	47
4.2	Questions constituting the HIT on Dialog Evaluation (Q: Question, Opt: Options). The questionnaire covers <i>the user's confidence, the perceived task completion, the expected behavior, the overall performance and the categorization of task success.</i>	52

4.3	Definitions of different levels of task success, based on the ITU Recommendation [2]. These definitions serve as the options to question 5 in the questionnaire.	53
4.4	An ambiguous dialog from TaskIncomplete category.	59
4.5	A TaskComplete dialog from the dialog set in the HIT on inter-rater agreement.	63
4.6	Expert-Workers and Expert-Expert agreement. There's a high level of agreement between experts and Workers for question 2 and 5 which are about classification of task completion, while the other questions about the overall user satisfaction have a lower agreement.	66
4.7	Comparison of labels from CMU experts and Workers.	68
5.1	Features automatically extracted from log files. Each dialog is represented by a vector f_i concatenating all the features.	75
5.2	A dialog example with a discourse that is the most typical of category A (the highly rated dialogs). The users provide one piece of information at a time, and they are guided by the system step by step.	82
5.3	An example dialog from system 1.	83
5.4	An example dialog from System 2.	85
5.5	An example dialog from the system 3.	85
5.6	An example dialog from the system 4.	86

Chapter 1

Introduction

A spoken dialog system (SDS) is a computer system which supports human-computer conversations in restricted domains. It integrates technologies including speech recognition, natural language understanding, dialog modeling, language generation, and text-to-speech synthesis. Advances in speech and language technologies have made SDSs an important research area and have brought about systems in a wide variety of application domains, such as flight information [3], bus schedule inquiries [4], stock market information delivery [5], tourist guide [6] and student tutoring [7]. To facilitate the development of SDSs and compare the performance of different SDSs, it is necessary to conduct SDS evaluation with appropriate methodologies, on which this thesis is focused.

In this chapter, we will briefly introduce the architecture of an SDS, especially the dialog model which is the principal component of an SDS. Additionally, we will cover SDS evaluation, a task that plays an important role in the system development.

1.1 SDS Architecture

An interaction between two participants is composed of many spoken dialog turns. A spoken dialog turn is a process in which one participant A says something to the other one B , and B interprets A 's dialog turn and then

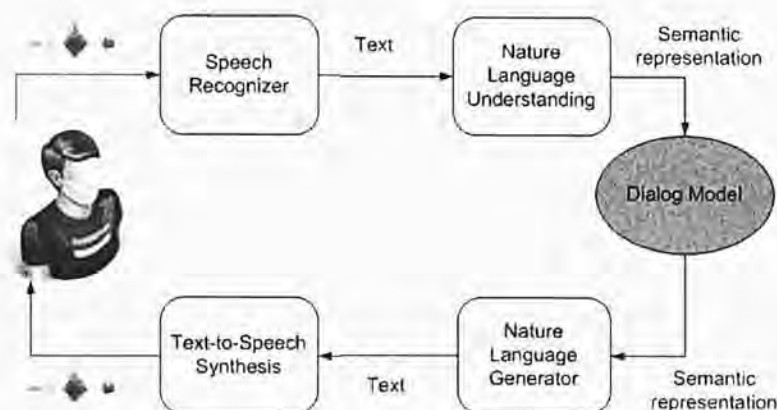


Figure 1.1: A typical spoken dialog turn between the system and the user

responds accordingly. A typical turn between the system and the user is shown in Figure 1.1.

As illustrated in Figure 1.1, the first step for the system is to recognize the user's speech and understand the underlying meaning. The speech is represented as a sequence of feature vectors (typically cepstral coefficients) and passed to the speech recognizer which usually employs a statistical acoustical model called hidden Markov model (HMM) [8] and a language model to produce the most probable hypothesis. State-of-the-art recognizers can produce a list of hypotheses called *N*-Best lists, or hypotheses embedded in a lattice.

A language understanding component then applies a lexicon of the language, a parser and a set of domain-specific grammar rules to analyze the hypothesis into an internal representation. The most important thing for the language understanding of a dialog system is to extract the user's intention and infer the appropriate actions according to the input sentence. Different kinds of language understanding components have been proposed, *e.g.*, belief networks are used in [9] [10] to infer the communicative goal of the user for information-seeking systems, and a flexible frame-based parser is used in [11] to analyze as much of the input as possible to provide accurate and robust responses to ungrammatical queries, *etc.*

The dialog model is the principal component of a dialog system which

maintains the history of the dialog, decides which action is appropriate based on language understanding, and controls the dialog flow. A more thorough introduction to the dialog model is provided in Section 1.2.

After the dialog model takes a proper action as the response, the next thing is to convey the system's response to the user in the audio form. The natural language generator [12] is responsible for translating the representation of the response semantics into text, which is then passed to the TTS synthesizer to generate audio output.

1.2 Dialog Model

The dialog model captures the way that human and the system converse, which involves in dialog states, state transitions and the dialog policy. Dialog states represent the results of performing system actions in previous states; state transitions promote dialogs to move forward; and the dialog policy is a functional mapping from dialog states to system actions. The dialog model chooses a system action at a certain dialog state according to the dialog policy, updates the dialog state based on the result of the previous action and controls the dialog to flow smoothly. Generally, dialog models are classified into three categories, *i.e.*, system-initiative, user-initiative and mixed-initiative.

The system-initiative dialog model is the simplest type in which the system is always in control to guide the dialog at each step and limits user's options to a small scale. Hence such dialog systems are able to work robustly by achieving a high level of task completion rate. The interactive voice response (IVR) system is a typical system-initiative system. It directs users in the dialogs to serve their inquiries by using the prerecorded audio [13] [14] [15]. One main disadvantage of such systems is the inflexibility and restrictions in the human-computer interaction which usually leads to a low level of user satisfaction.

On the contrary, the user-initiative dialog model allows users to completely control the dialog flow. In user-initiative dialogs, the system is required to respond sufficiently whatever users request. However, this ideal strategy cannot be reached with existing technologies.

The mixed-initiative dialog model allows both users and the system to involve in the control of the dialog flow. The user can barge in and change the dialog direction at appropriate times. The system follows the user's request and can direct the user when he/she goes off the dialog course. The mixed-initiative dialog model is flexible, accommodates imperfections of existing using technologies and is the most commonly used in current dialog systems [7] [5] [4].

The partially observable Markov decision process (POMDP) has been widely applied to the mixed-initiative dialog modeling by many researchers recently [16] [17] [18] [19]. It has the advantages of controlling the dialog flow in SDSs in the aspects of handling uncertainties due to speech recognition and trading off the effects of different system actions. First, POMDP-based SDSs represent the user state (including the user's true intention, the user's true actions, *etc.*) as an unobservable POMDP state, and maintain a probability distribution over all the possible states based on the observations (i.e., the recognition results). Second, a POMDP assigns a reward for taking a certain action in a certain state, and chooses an action by maximizing the cumulative sum of rewards over the temporal process. In this thesis, we implement a bus information dialog system based on POMDP.

1.3 SDS Evaluation

As SDSs are becoming increasingly pervasive in supporting information access by the mass, its diversity calls for sound strategies in evaluating, comparing, and predicting the performance of SDSs. Therefore, developing principled ways of evaluating an SDS becomes a hot research area recently. Generally, SDS

evaluation can be categorized into *component-based* perspective and *holistic* perspective.

The component-based perspective covers the performance of individual components such as the correctness in speech recognition, the ability to understand natural language, the appropriateness in response generation, as well as the naturalness of the synthetic speech in conveying the responses. A thorough evaluation of an SDS needs to consider all relevant evaluation metrics covering the functionalities for all the system components [20].

In contrast, holistic evaluation, which assesses not only individual components but also the integrate performance of an SDS, is more appropriate. It involves the perceived level of system usability, system intelligence, and abilities in error recovery by considering the system entirely [21]. It also needs to cover the wide variety of users' impressions (user judgments) relating to all dimensions of the quality of an SDS [22]. The ultimate objective of an SDS is to satisfy the demands of real users. Therefore, user satisfaction (how much the user is satisfied with the system performance) is considered the most important criterion for the system performance among all the user impressions [23].

Many evaluation methods have been developed in recent years. One popular method of measuring user judgments is to invite subjects to fill out a questionnaire after they interact with an SDS. The questionnaire often involves all aspects of perceptions on the system such as task completion and user satisfaction. In spite of its popularity, this traditional way has some disadvantages. First, it is a costly and time-consuming process. Moreover, due to constrained resources, this approach is often limited to a small number of evaluators whose feedbacks may not be statistically representative of the large user population that can access the SDS. Furthermore, in some situations where the system has already been in deployment, it is often difficult to ask real users to patiently complete an evaluation survey.

Another popular evaluation method, the PARADISE framework, has been

proposed for automatic inference of overall user satisfaction of unrated dialogs [24]. It assumes that the overall performance (user satisfaction) of an SDS can be described in terms of a linear regression model of a set of dialog metrics [25]. The trained model can explicitly demonstrate which factors have significant contributions to user satisfaction. However, its predictive power is limited, especially on test data, *e.g.*, R-squared (R^2)¹ around 0.22 on the dialog corpus in [26]. Low R^2 may be caused by the lack of inter-rater agreement on user satisfaction ratings [27], or the linear model may be insufficient in capturing the relations between user satisfaction and dialog features.

The primary emphasis of this thesis is placed on developing an efficient and effective paradigm for SDS evaluation. It has been noticed that crowdsourcing technologies are widely applied by many researchers to collect, transcribe and annotate speech and language data in recent years [28] [29] [30]. Crowdsourcing refers to outsourcing a task to a crowd of people. Unlike using the traditional method in which data is manually labeled by experts or trained people, tasks can be completed with crowdsourcing in a cost-effective, efficient and flexible manner. Novotney *et al.* collected high quality transcriptions of conversational speech with only one thirtieth the cost of professional transcription [30]. Snow *et al.* conducted varieties of NLP tasks in the crowd [29]. We feel that user judgments for SDS evaluation can also be collected by using crowdsourcing instead of user experiments. In this thesis, we develop a crowdsourcing methodology for this purpose through Amazon Mechanical Turk (MTurk)². Collaborative filtering (CF) has been successfully applied to the development of recommendation systems [31]. It assumes that the preference of a user for a new item may

¹ R^2 is a statistical measure of how well a regression model approximates real data points; an R^2 of 1.0 (100%) indicates a perfect fit. The formula for R^2 is: $R^2 = 1 - \frac{\sum_{i=1}^n (r_i - \hat{r}_i)^2}{\sum_{i=1}^n (r_i - \bar{r})^2}$, where r_i is the true rating, \hat{r}_i is the predicted rating from a prediction model, and \bar{r} is the mean of $\{r_i\}_{i=1}^n$.

²www.mturk.com. MTurk is a popular crowdsourcing marketplace that makes use of human intelligence online to perform tasks which cannot be completed entirely by computer programs.

resemble that for the similar items rated previously, which also holds for automatic evaluation of SDSs, *i.e.*, the rating of an unrated dialog is similar to those of its neighbors. Therefore, this thesis extends the PARADISE framework by incorporating CF to improve its prediction performance, based on the collected user judgments through MTurk.

1.4 Thesis Outline

In Chapter 2, we review the literature of approaches to dialog modeling, and provide a brief overview of related work on SDS evaluation. Chapter 3 describes the implementation of a bus information dialog system based on the SDS-POMDP model proposed in [18]. In the later work, the performance of this POMDP-based system will be evaluated by our evaluation paradigm for SDSs which will be detailed in Chapter 4 and Chapter 5.

Chapter 4 presents an initial attempt at the use of crowdsourcing for collecting user judgments on an SDS through MTurk. We design two types of tasks – the first targets at fast rating a large number of dialogs regarding some dimensions of the SDS’s performance and the second aims to assess the reliability of Workers on MTurk. A set of approval rules are also designed to control the quality of ratings from MTurk. Analysis of collected results demonstrates that the crowdsourcing method for the collection of user judgments is efficient, flexible and cost-effective. The comparison of annotations between experts and MTurk Workers on SDSs shows a high level of agreement between the two groups, which supports the reliability of crowdsourcing.

Chapter 5 proposes a collaborative filtering model to improve the accuracy of predicting user satisfaction in SDS evaluation. This prediction model is drawn from the idea of CF in recommendation systems, which uses information from near neighbors of an unrated dialog to predict its user satisfaction. Experimentation is conducted based on the collected user judgments in Chapter

4, and results show that the CF approaches distinctly improve the prediction accuracy. The CF model is generalizable to multiple SDSs within bus information domain. It is also applied to evaluate the performance of the bus information system in Chapter 3. Finally, this thesis ends with Chapter 6.

Chapter 2

Previous Work

This chapter first reviews the previous literature in dialog modeling. Section 2.2 then provides a brief overview of metrics for SDS evaluation. It ends with the introduction to the PARADISE framework which has been widely applied for SDS evaluation.

2.1 Approaches to Dialog Modeling

The introductory chapter has stated that the dialog model is the principal component in the dialog system and takes charges of controlling the dialog flow. The approaches to dialog modeling can be summarized into two categories, *i.e.*, handcrafted and statistical.

2.1.1 Handcrafted Dialog Modeling

Traditionally, dialog models are manually devised by system designers, which explicitly express dialog states, state transitions and the dialog policy. There are four approaches to handcrafted dialog models, *i.e.*, finite state machine (FSM) based, form-based, agenda-based and information-state-based approaches.

1. FSM-based approaches

Early approaches to dialog modeling are based on FSM. The nodes in an FSM represent the dialog states, and each node is associated with a system action. The state transition between two nodes depends on the user's response to the previous action of the system. Thus, all possible dialogs are specified by the paths in the network [32]. System designers should define the whole network including the nodes, state transitions and system actions. Telephone-service systems are typically realized by using FSM for dialog modeling [33]. Some toolkits like CSLU have also been developed for constructing FSM-based dialog systems of different types [34].

The advantage of FSM-based systems is that they work straightforwardly, robustly and effectively. The disadvantages are that they are inflexible and are mainly used for system-initiative dialog modeling. They put many constraints on the human-computer interaction, and the user must follow the predefined path. If the user changes his goal or deviates from the original path, the dialog can hardly get into the right track again. This limitation promotes approaches to the mixed-initiative dialog modeling.

2. Form-based approaches

The form-based approach also called slot-filling is commonly used for the mixed-initiative dialog modeling. This approach represents the data structure as a form with several slots which are related to the concepts of domains. For example, in the bus information domain, there are slots for the destination, the origin, the leaving time, the arrival time and the bus route. Each slot has a set of values and is associated with a set of predefined system actions. The slots filled with different slot values form dialog states. Slots, slot values and system actions are designed by system developers according to the specific application domain. The

dialog proceeds by taking an appropriate system action at a certain dialog state according to the handcrafted rules (the dialog policy). Goddeau *et al.* developed a form-based dialog system in the WHEELS domain which provides thousands of advertisements of used automobiles [35]. This WHEELS system requires users to explore the combination of slots rather than simply ask them to fill out slots one by one, which enables the mixed-initiative interactions.

Form-based dialog systems relax the constraints on the users so that users can change task goals at proper times. However, in some complex applications (*e.g.*, traveling planning) where the user's goal is not fixed, it is difficult for form-based systems to fill out slots successively.

3. Agenda-based approaches

Agenda-based dialog models can accommodate more complex circumstances. Since many complex dialog tasks can be treated as a succession of topics, they are often decomposed as a hierarchical tree of dialog agents. In an agenda-based system, designers also need to handcraft an agent tree to specify the dialog task in addition to the dialogs states and state transitions. In the RavenClaw architecture [1], the layer for the dialog task specification is represented as a tree of agents shown in Figure 2.1. In this tree, the root node decomposes into subtasks, *i.e.*, Login (verifying the user's identity to the system), GetQuery (obtaining the time, the location, and other information from the user), GetResults (searching for results in the database according to the query), *etc.* Further, GetQuery subsumes Time and Location, which obtains the time and the location information from the user. Examples also include a tree of handlers (a list of topics) representing the agenda in Agenda Communicator [36].

The agenda-based approach is suitable for *information-querying* domains where the user tries to access certain information, such as travel planing.

tourists guide, *etc.* However, in domains (*e.g.*, tutoring) where what the user wants cannot be determined, these approaches often have trouble in modeling interactions successfully.

4. Information-state-based approaches

The Information-state approach proposed by Larsson *et al.* views the functions of the dialog model in terms of an information state which identifies relevant aspects of the dialog [37]. The information state represents the accumulation of all previous information in the course of dialogs and motivates the next system action. It is different from the dialog state which is used in the three kinds of approaches above. The dialog state represents the results of performing dialog actions in previous states. It is hard to transform an information state to a dialog state, while the dialog state itself can be viewed as an information state. In order to build a complete information-state-based dialog system, we need to design a formal representation of the state, a set of dialog moves (system actions), a set of update rules (state transitions) and an update strategy (the dialog policy). TrindiKit is a popular toolkit for developing information-state based dialog systems [37]. Bos *et al.* constructed the DIPPER dialog model by improving the update rules in TrindiKit [38].

2.1.2 Statistical Dialog Modeling

In handcrafting dialog models, designers need to consider all possible situations and test dialog models iteratively until all conditions have been covered, which is a time-consuming process. When applications become complex, systems will encounter some unforeseen situations and the handcrafted approaches become unsatisfied. An alternative to the mixed-initiative dialog modeling is to apply stochastic process models. In the statistical approach, a dialog is modeled

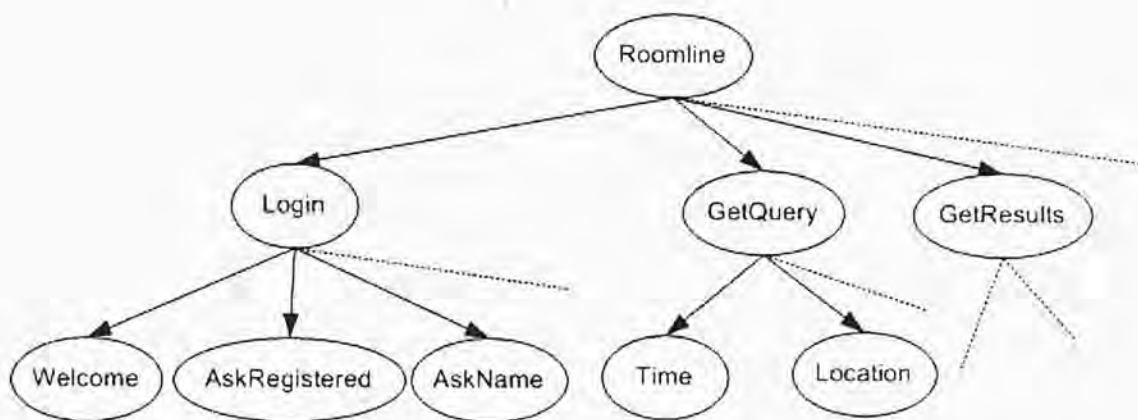


Figure 2.1: A part of the tree of agents in the RavenClaw architecture [1].

as a stochastic process in which the state transition is not determined by designed rules but occurs with a certain probability. The stochastic model is characterized with a set of parameters (*e.g.*, the transition probabilities) which can be automatically learned from a training set. Two popular statistical approaches are the Markov decision process (MDP) and POMDP.

1. The MDP approach

The MDP model serves as a formal representation of the human-computer interaction and provides the basis for formulating the dialog policy learning problems. The MDP dialog model is formalized with the state space relating to dialog states, system actions and a reasonable dialog policy. In a MDP, the system will get a reward or pay a cost by taking an action at a certain dialog state. An optimal dialog policy maps from dialog states to system actions, and is automatically learned by maximizing the expected cumulative sum of rewards or minimizing the expected cumulative sum of costs. In this way, the optimal dialog policy can trade off well between the effects of different system actions at a certain dialog state. Levin *et al.* constructed a MDP system in the air travel information system (ATIS) domain [39]. This system needs to pay a cost when asking users for flight information or accessing a database,

and its dialog policy is learned by minimizing the expected cumulative costs of these actions. Walker *et al.* used the expected rewards in a MDP-based system to compare the performance of system-initiative and mixed-initiative styles [40].

MDPs are useful for generating dialog policies. However, they cannot handle uncertainties well, since they assume the current dialog state is known exactly and the optimal policy is directly expressed by a function mapping from dialog states to system actions. The conversation state in a dialog system contains uncertainties since both the speech recognizer and the language understanding are error-prone, especially in noisy environments. As the uncertainties increase, the performance of MDP-based dialog systems degrades.

2. POMDP approaches

The POMDP is an elegant mathematical model to handle uncertainties and have been applied to dialog modeling by many researchers [16] [17] [18] [19]. It is a general form of the MDP. Unlike the MDP whose hidden state represents a certain dialog state, the hidden state in POMDP represents a probability distribution over all the possible dialog states. Roy *et al.* first proposed to use POMDP for the generation of dialog policies by representing the hidden state with the user's intention [41] [17]. Williams *et al.* factored the POMDP state into three components according to the characteristics of SDSs, *i.e.*, the user's goal, the user action and the dialog history, and formalized the SDS-POMDP model for building an SDS [18]. All of their work has demonstrated that the POMDP-based dialog system outperforms the MDP-based one in the same noise condition.

There are two essential components in POMDP-based dialog models, *i.e.*, maintaining the probability distribution over all the possible dialog

states called belief state and finding the optimal dialog policy which maximizes the cumulative sum of rewards over time. Previous work on POMDP-based dialog modeling is summarized in the two aspects.

(a) Policy learning

Although the POMDP model is suitable for handling uncertainties, it faces severe scalability challenges. Adding more dialog states results in more computations for policy learning, and even leads to the intractability of optimization. The tractability problem prevents POMDPs from being applied to construct real systems which often require a large number of dialog states. For example, the system by Roy *et al.* [17] only includes 13 dialog states and the early system by William *et al.* [18] is limited to only 6 user goals. Much research work is devoted to the tractability problem.

Williams *et al.* proposed the composite summary point-based value iteration (CSPBVI) to perform optimization for POMDP-based dialog systems of a realistic size [42]. The key idea of this algorithm is to perform optimization for the summary POMDPs of 6 states. The summary-level actions then are mapped to the master-level ones through the defined heuristics. This algorithm has been applied in this thesis to implement a dialog system in bus information domain.

Bui *et al.* presented a dynamic decision network POMDP (DDN-POMDP) approach to handle a large number of slots [43]. The DDN-POMDP dialog model is composed of two sub-models, *i.e.*, the slot-level and the global dialog models. The slot-level dialog model is formulated as a POMDP and its policy learning is realized through k -step look-ahead DDNs. The global dialog model is responsible for choosing system actions through heuristics by aggregating the nominated slot-level actions. Young *et al.* used the hidden information

state (HIS) model to build a practical system for the tourist information domain [44]. In the HIS-POMDP model, equivalent dialog states are grouped into the same classes called partitions, and the root partition is further split into smaller ones as dialog progresses. Through state partitions, much computation is saved.

(b) Belief state maintaining

Belief state update is also computationally expensive when the number of states is large. Updating belief state efficiently is important for building a dialog system serving for real users, since the delay caused by the real-time belief updating has a large impact on the system performance. Thomson *et al.* introduced an approximate method for efficient updating the belief state called the Bayesian Update of Dialog State (BUDS) [45]. Besides representing the hidden state with the user goal, user action and dialog history, BUDS further factorizes each component according to the concepts of each slot. Through the combination with the factor graphs and the loopy belief propagation algorithm, the computation efficiency is significantly improved.

2.2 Evaluation Metrics

As introduced in Chapter 1, SDS evaluation plays a critically important role in the development of successful SDSs. The performance of an SDS can be measured with many aspects of metrics, such as task success, the number of utterance, the speech recognition accuracy, the system response delay, the naturalness of the output speech, the user's expectation and cooperativeness of the system [20]. These metrics for both component-based and holistic evaluation are usually categorized into *subjective* and *objective* metrics. Subjective metrics, which reflect users' perceptions on the quality of an SDS, are often

obtained from real or test users. Objective metrics, which quantify the system behavior during the interaction and the performance of the components of an SDS, can be extracted automatically or labeled manually by experts from the user-system interactions. The objective metrics are also called interaction metrics in [20].

2.2.1 Subjective User Judgments

Since the subjective metrics mostly rely on user judgments on the system quality, distributing questionnaires to users before or after the interaction with an SDS is an effective way to collect quantifiable user judgments.

Developing a reliable and valid questionnaire for subjective judgments collection attracts many researchers' attentions. The SASSI questionnaire (Subjective Assessment of Speech System Interfaces) is designed for subjective assessment of speech-based systems [46]. SASSI consists of 50 items (statements), and each item is rated by users on a 7-point scale agreement: strongly agree, agree, slightly agree, neutral, slightly disagree, disagree and strongly disagree. A factor analysis of the collected data from 226 completed questionnaires suggests that six main factors contribute to a user's subjective perceptions on speech-based systems, *i.e.*, perceived system response accuracy, likeability, cognitive demand, annoyance, habitability and speed.

The ITU recommendation proposed another list of questions for the evaluation of telephone services based SDSs [47]. Three types of questionnaires are distinguished in the recommendation. Type 1 questionnaires are intended to collect user's background information and are distributed at the beginning of an evaluation experiment. Type 2 questionnaires are related to the user-system interactions. The last type questionnaires are about users' overall impression of the system quality. A list of topics are proposed for each type of questionnaires and the exemplary statements are rated on a 5-point scale.

2.2.2 Interaction Metrics

In contrast to the subjective judgments on system performance, interaction metrics can easily quantify the ability of the system or components to perform the designed functions. Such information is obtained from the log files which record the interactions between the system and users. Some metrics about the surface of the system/user utterances can often be automatically extracted from the log files, such as dialog duration, or recognition confidence score, while others which are related to the content of the interaction are usually manually labeled by experts or trained people, like the understanding accuracy, or task success.

In recent decades, many metrics have been identified to measure the functions of system or its components. Early metrics are for individual components, such as speech recognizer and language understanding component. The commonly used metrics are *Word Accuracy (WA)*, *Sentence Accuracy (SA)*, *Concept Accuracy (CA)*, *Query Density (QD)*, *Concept Efficiency (CE)* etc. Later, metrics for the whole systems have been developed, including *Task Success (TS)* to measure the extent to which the system achieves the task, *number of dialog turns* for measuring the dialog cost, or *Contextual Appropriateness (CA)* for measuring the degree to which the system provides an appropriate response [48] [34].

Based on the literature of interaction metrics, Möller *et al.* summarize a set of metrics for SDSs evaluation and classify them into five categories:

- Dialogue- and communication-related category: Metrics about the overall dialogue, such as overall dialog duration, dialog turns, or average number of words per system turn during the dialog, *etc.*
- Meta-communication-related category: Metrics describing the recognition and understanding capabilities, such as number of help requests, number of barge-in attempts from the user, *etc.*

- Cooperativity-related category: Metrics about the cooperativity of system actions (responses). The contextual appropriateness of system responses directly measures cooperativity, which is often judged by several experts based on Grice's maxims principle.
- Task-related category: Task success is a key aspect of successful task-oriented systems. Möller defined task success in seven aspects, *i.e.*, succeed by providing a completely right answer; success with constraints from user, or from system, or from both user and system; succeed by spotting no solutions exist; failure resulting from user's non-cooperative behavior or system's inappropriate response.
- Speech-input-related category: Metrics about the capability of systems to recognize the input speech and to understand the meaning of inputs. Common used metrics are *WA*, *SA*, or *CA* as introduced above.

This categorization and the metrics in each category have been incorporated in the ITU recommendation [2].

2.3 The PARADISE Framework

PARADISE (PARAdigm for Dialogue System Evaluation) is a general framework for evaluating and comparing the performance of spoken dialog systems [24]. It specifies which system property has a large impact on system usability and supports the development of predictive models of system performance.

PARADISE uses decision theory methods to relate a battery of dialog metrics to the system's overall performance and determine the significant contributors. The PARADISE performance model is shown in Figure 2.2. In this model, the overall performance is correlated with user satisfaction and the primary objective of a system is to maximize user satisfaction. This objective can be further decoupled into two sub-objectives: maximizing task success and

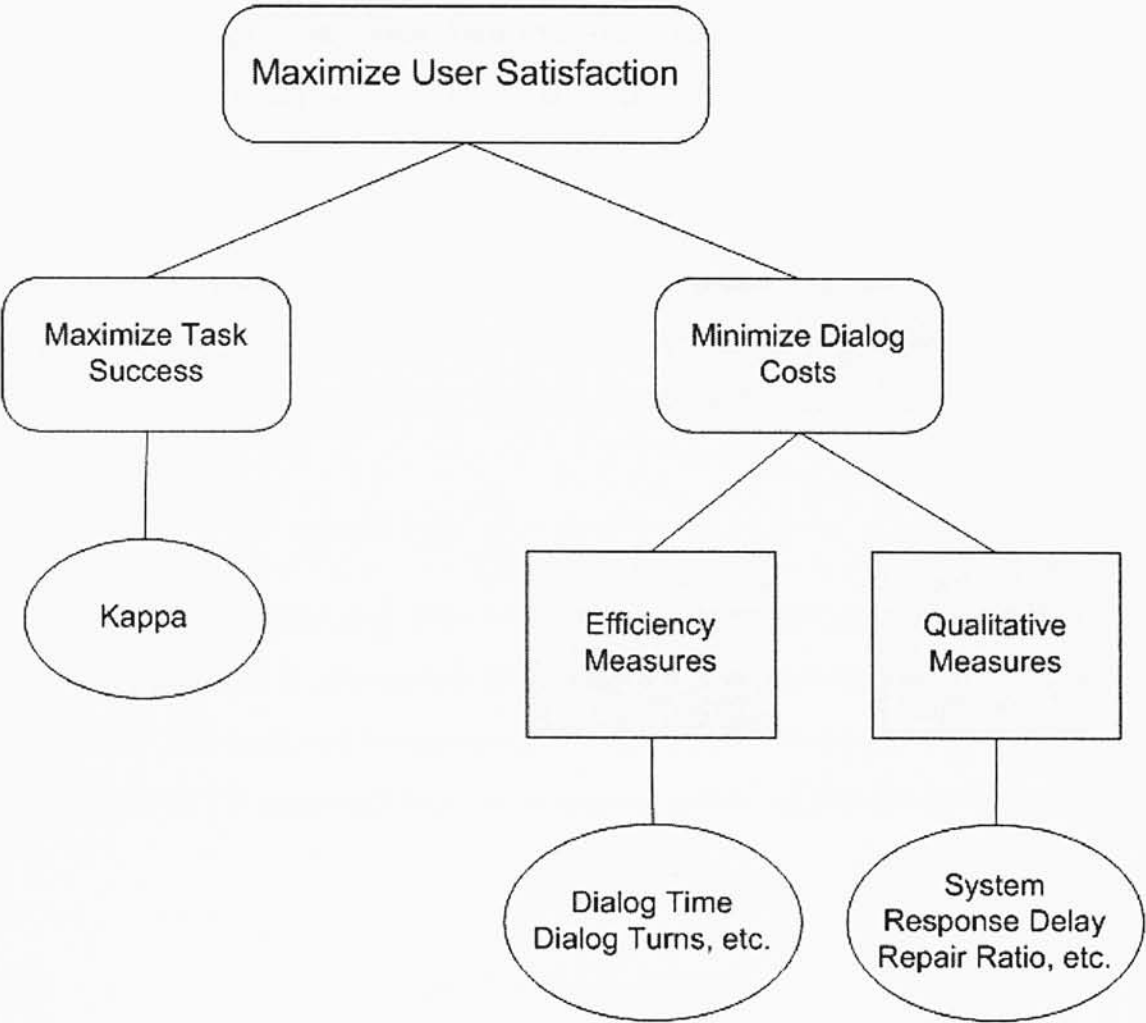


Figure 2.2: The PARADISE structure of objectives for spoken dialog performance

minimizing dialog cost, based on the assumption that task success and dialog cost are two main types of contributors to user satisfaction. In the original PARADISE framework, task success is measured with the use of the Kappa coefficient and attribute value matrix (AVM). Dialog costs can be categorized into two types: dialog efficiency and quality. Dialog efficiency is represented by dialog turns or dialog duration, while dialog quality is measured in terms of the appropriateness of system response, or system repairing ratio, *etc.*

The PARADISE framework posits that the objective structure in Figure 2.2 can be realized by building a performance model through multivariate linear

regression with user satisfaction as the target and the dialog metrics of task success, dialog efficiency and quality as predictors. Building the performance model requires a dialog corpus be collected through controlled user experiments during which users subjectively rate their satisfaction. Moreover, the predictors of the model, *i.e.*, the dialog metrics, can be either automatically extracted from dialog log files or manually labeled by experts. Based on these illustrations, the performance model of an SDS is defined as below:

$$S_u = (\alpha * N(\kappa)) - \sum_{i=1}^n w_i * N(c_i), \quad (2.1)$$

where S_u is system performance correlated with user satisfaction here, κ is a measure for task success, c_i is a measure for dialog cost, α is a weight on κ , w_i is a weight on c_i , and $N(\cdot)$ is a z -score normalization function [49]. Both κ and c_i can be represented as dialog measures m , and Equation 2.1 is transformed into a simpler one:

$$S_u = \sum w_i * N(m_i). \quad (2.2)$$

Since the dialog measures have been normalized into the same scale by $N(\cdot)$, the weight w_i reflects the relative contribution of the corresponding measure m_i to user satisfaction.

By applying the performance model, values of user satisfaction of SDSs are directly predicted from a suite of dialog metrics which are simply extracted from dialogs, without the need of conducting user experiments to assess user satisfaction. In addition, system developers can directly figure out which system components have larger impact on user satisfaction by observing the coefficients of dialog metrics in the performance model, so that they can focus on improving the performance of those “important” components. In this way, an efficient system design process is ensured, *i.e.*, focusing on high service quality for the end user.

PARADISE has been widely applied in evaluating many SDSs, such as the ITSPOKE tutoring system [50] and DARPA Communicator [51]. It is applied to test different dialog models with two systems ELVIS and TOOT in [25]. The ELVIS experiments tested different initiative dialog models, while the TOOT experiments tested models with different information presentation policies. Regression functions are derived for each model to analyze their performance. Kamm *et al.* discusses the generalization ability of PARADISE across three different systems [52]. Walker *et al.* study its generalization across different user population and find that it does not generalize well from novice users to expert users [53]. Additionally, researchers extend PARADISE to evaluate multimodal systems [54] [55].

2.4 Chapter Summary

This chapter first reviews the literature of dialog modeling which is essential for controlling dialog flow during human-system interaction. In addition, we present the common evaluation metrics which are related to our work on SDS evaluation in Chapter 4 and 5. Finally the PARADISE framework for SDS evaluation is described. It is extended to improve the accuracy of predicting SDS performance by introducing collaborative filtering in Chapter 5.

Chapter 3

Implementation of a Dialog System based on POMDP

As introduced in Chapter 2, there are many uncertainties existing in spoken dialog modeling, such as the uncertainty about the user’s words recognized by the error-prone automatic speech recognition (ASR), or the uncertainty about the user’s intention interpreted by the understanding component. These uncertainties bring about difficulty for dialog modeling. Many researchers have applied partially observable Markov decision processes (POMDPs) to dialog modeling which takes into account the uncertainties when planning, and have demonstrated that POMDP-based systems work robustly in noisy environment. Before introducing the proposed evaluation paradigm for SDSs, we first describe the implementation of a POMDP-based dialog system in this chapter, whose performance will be assessed by our evaluation methodology later.

This chapter first introduces the formulation and planning of POMDPs, associated with a toy example for illustration. Then it describes a statistical model of SDS-POMDP for dialog modeling proposed in [18], and the corresponding optimization algorithm—composite summary point-based value iteration (CSPBVI)—for building SDS-POMDP based dialog models of a realistic size [42]. In the end of this chapter, we develop a working dialog system in bus information

domain based on SDS-POMDP optimized with CSPBVI.

3.1 Partially Observable Markov Decision Processes (POMDPs)

3.1.1 Formal Definition

A POMDP is defined as a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{O}, \mathbb{T}, \mathbb{R}, \Omega, \gamma)$, where

- \mathbb{S} is a set of states $\{s_1, s_2, \dots, s_n, \dots\}$,
- \mathbb{A} is a set of actions $\{a_1, a_2, \dots, a_n, \dots\}$,
- \mathbb{O} is a set of observations $\{o_1, o_2, \dots, o_n, \dots\}$,
- \mathbb{T} is a set of conditional transition probabilities: $P(s'|s, a)$,
- \mathbb{R} is a reward function: $\mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$,
- Ω is a set of conditional observation probabilities: $P(o'|s', a)$,
- γ is a discount factor, $\gamma \in [0, 1]$. It is used to keep a finite sum for infinite horizon.

A defined POMDP models the interaction between an agent and the environment as the iterations of the following steps:

1. At each time step, the environment is in certain unobservable state $s \in \mathbb{S}$, and a distribution over all the possible states is maintained. This distribution is called belief state $b(s)$. Based on the state distribution b , the agent takes an action $a \in \mathbb{A}$ and it will receive an immediate reward $r(s, a)$. After, the environment will transit to another unobservable state s' according to the transition probability $P(s'|s, a)$.

2. The agent will receive an observation $o' \in \mathbb{O}$ from the environment based on the state s' and its action a . At this time the belief state $b(s)$ can be updated to $b(s')$ with the transition model $P(s'|s, a)$ and the observation model $P(o'|s', a)$.

The belief state $b(s)$ at each time step summarizes all the information in previous time steps. By applying the Bayesian rule, the belief could be updated as follows:

$$\begin{aligned} b_a^o(s') &= \frac{P(o'|s', a) \sum_s (P(s'|s, a) b(s))}{P(o'|a, b)} \\ &\propto P(o'|s', a) \sum_s (P(s'|s, a) b(s)). \end{aligned} \quad (3.1)$$

As mentioned above, the agent will receive an immediate reward $r_t(s, a_t)$ at time step t after taking the action a_t . The corresponding immediate reward of belief state can be calculated as the expectation of the rewards over all the states,

$$r_t(b_t, a_t) = \sum_s b_t(s) r_t(s, a_t). \quad (3.2)$$

The cumulative, expected discounted reward by time step t denoted as V_t is calculated as follows:

$$\begin{aligned} V_t &= \gamma^0 r_t + \gamma^1 r_{t-1} + \gamma^2 r_{t-2} + \cdots + \gamma^{t-1} r_1 + \gamma^t r_0 = r_t + \gamma V_{t-1} \\ &= \sum_s b_t(s) r(s, a_t) + \gamma \sum_s b_t(s) \sum_{s'} P(s'|s, a_t) \sum_{o'} P(o'|s', a) V_{t-1}(b_a^o). \end{aligned} \quad (3.3)$$

As observed in Equation 3.3, the expected value V_t is related to b_τ and a_τ , where $\tau = 0, 1, \dots, t$, indicating that different actions lead to different belief states and different immediate rewards, and finally result in different expected values. Hence the planning problem for the interaction between the agent and the environment with a defined POMDP has been converted into the problem of finding the optimal policy, *i.e.*, choosing actions at time t so as to maximize the cumulative expected value V_t . An optimal policy denoted as π^* can be expressed as follows,

$$\pi_t^*(b) = \arg \max_a \left[\sum_s b(s) r(s, a) + \gamma \sum_s b(s) \sum_{s'} P(s'|s, a) \sum_{o'} P(o'|s', a) V_{t-1}(b_a^o) \right], \quad (3.4)$$

where $a \in \mathbb{A}$. The value of the optimal policy denoted as V^* or V_{π^*} is achieved by the Bellman backup operator H ,

$$\begin{aligned} V_t^*(b) &= \max_a \left[\sum_s b(s) r(s, a) + \gamma \sum_s b(s) \sum_{s'} P(s'|s, a) \sum_{o'} P(o'|s', a) V_{t-1}(b_a^o) \right] \\ &= HV_{t-1}(b). \end{aligned} \quad (3.5)$$

Each policy can be represented by a finite set of value vectors in the $|S| - 1$ dimension belief space $\{\alpha_t^1, \alpha_t^2, \dots\}$ and each vector is associated with an action. Hence, the optimal policy π^* is represented as convex piecewise lines in belief space. Once π^* of a defined POMDP is learned, the agent makes a decision based on π^* when it is staying belief state $b(s)$. This decision is optimal since it maximizes the cumulative value by current time.

3.1.2 Value Iteration

Markov property allows exploitation of dynamic programming principle for optimal policy construction. Value iteration is an exact dynamic programming process for solving POMDPs. This method proceeds to find the optimal policy by applying Bellman backup operator to the optimal policy in the previous time step, given the initial value vectors. This algorithm operates as below,

1. $V_0(s) = 0, s \in \mathbb{S}$
2. $V_t^*(b) = \max_a \left[\sum_s b(s) r(s, a) + \gamma \sum_s b(s) \sum_{s'} P(s'|s, a) \cdot \sum_{o'} P(o'|s', a) V_{t-1}^*(b_a^o) \right]$
3. $\pi_t^*(b) = \arg \max_a \left[\sum_s b(s) r(s, a) + \gamma \sum_s b(s) \sum_{s'} P(s'|s, a) \sum_{o'} P(o'|s', a) V_{t-1}^*(b_a^o) \right]$

In POMDP, the belief state represents the probability distribution over all the possible states, indicating that the belief space is continuous. In the value iteration algorithm, the *max* action in the optimal policy generation step is taken over the whole belief space, which requires to solve multitude of linear programs. Due to the computational complexity, the exact POMDP optimization problem can only be limited to a small number of environment states.

3.1.3 Point-based Value Iteration

An alternative to the exact optimization technique is applying approximate techniques. A popular approximate algorithm is called point-based value iteration (PBVI). Instead of searching the whole belief space, PBVI finds the optimal policy only for a finite set of belief points sampled from the belief simplex. It is based on the assumption that the belief simplex is usually sparse and only a small number of belief points (compared to the continuous space) could be reached when the agent is interacting with the environment. As the number of belief points increases, the policy quality is increasing and so is the computation time.

3.1.4 A Toy Example of POMDP: The NaiveBusInfo System

In order to illustrate the basic operations and essential properties of POMDPs and how POMDPs are used in an SDS, we present a simply example concerning a simple application called the NaiveBusInfo.

In this example, NaiveBusInfo system provides two bus routes for users, namely *54C* and *61A*. The user tells the system which bus he/she wants to take (referring to the user's goal *s*). The machine has three available actions *a*: *ask* the user which bus to take, and *submit 54C* or *61A* to the database to

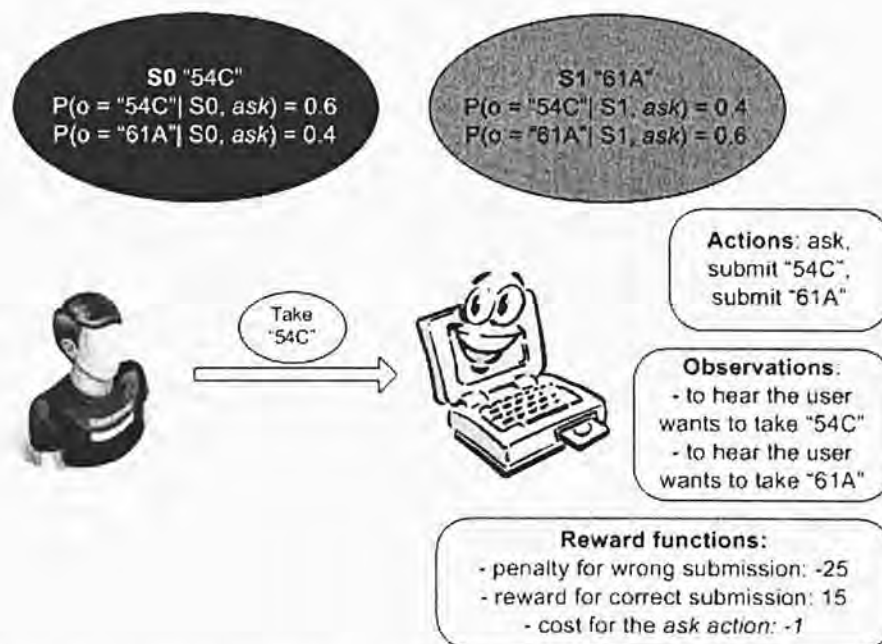


Figure 3.1: A POMDP example in the NaiveBusInfo system.

search for corresponding timetable information. As illustrated in Figure 3.1, when the system asks the user for the bus route, it receives correct information from the user with the observation probability $P(o = s | s, ask) = 0.6$, where $s \in \{s_0 = 54C, s_1 = 61A\}$. The system needs to take an appropriate action based on what is heard from the user (the observation o) and different system actions bring the system with different levels of rewards (negative or positive).

Since the system does not know at any time what the user's true goal is, it maintains a probability distribution over all possible states which is called the belief state b . From the knowledge of the transition probability $P(s_t | s_{t-1})$ and the observation probability $P(o_t | s_t)$, a new belief state is inferred through Bayes's rule. This process is called belief updating. Figure 3.2 shows the process of belief updating. In the time step $t - 1$, the belief of $54C$ and $61A$ is 0.5 and the system takes the action of *ask*. In the next time step t , the system receives the observation of $54C$ from the user, which results in a probability mass is shifting from the state of "61A" to that of "54C".

Based on the belief state at current time step, the system should carefully choose an action in response to the user. In the situation of Figure 3.2 where

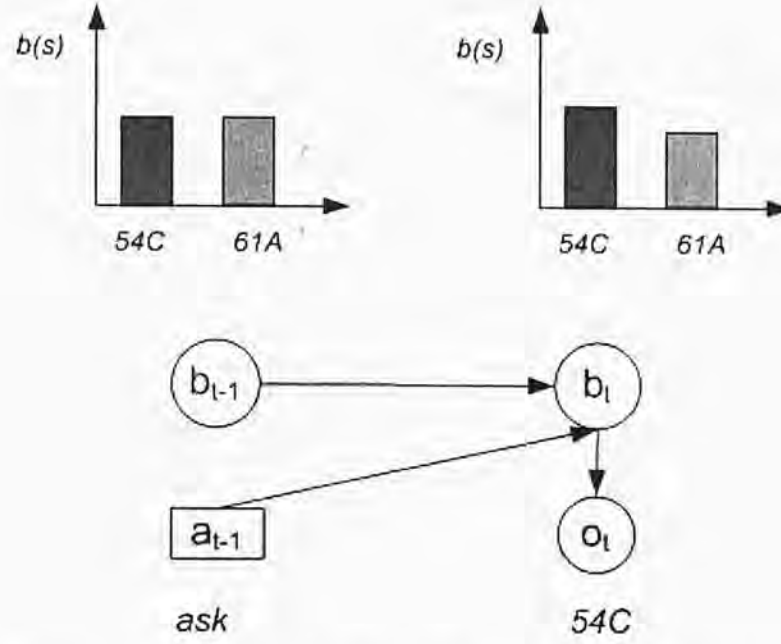


Figure 3.2: The process of belief updating based on the observation and the previous action of the system.

the probability of *54C* is a little higher than *61A*, the system can directly submit *54C* to the database by risking losing 25 reward (see Figure 3.1), or it can ask the user again to confirm this state by losing only 1 reward. Since different actions result in different rewards, the problem of designing the dialog strategy evolves into that of optimizing decisions in POMDP. The designing objective is to find a policy by maximizing the accumulative sum of rewards over time. The optimal policy π mapping from belief states to system actions is described in Figure 3.3.

For the NaiveBusInfo system, its optimal policy is expressed by piece-wise lines shown by the heavy lines in Figure 3.4. The policy divides the belief space into 3 partitions each of which is associated with an optimal action. We can observe that in the region of belief space close to the corners where certainty is high, the system chooses *submit 54C* or *61A*; and in the central region where certainty is low, the system chooses the *ask* action. When the system executes this policy, actions are selected depending on the partition which contains the current belief state. Take the two belief states in Figure

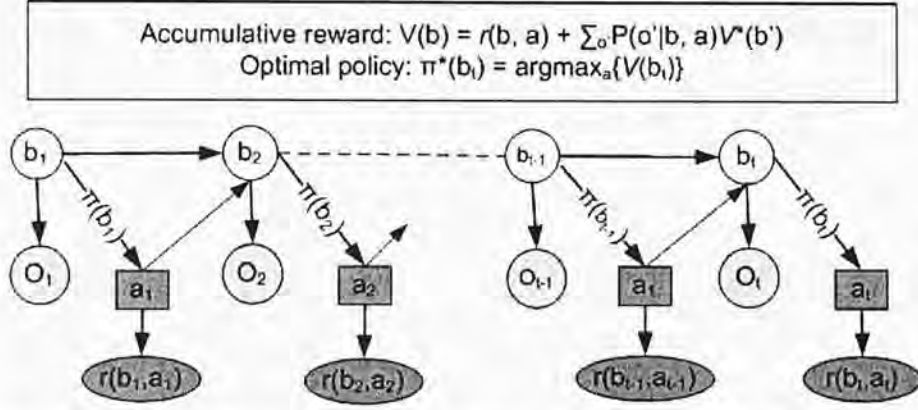


Figure 3.3: Illustration of the optimal policy in a POMDP framework.

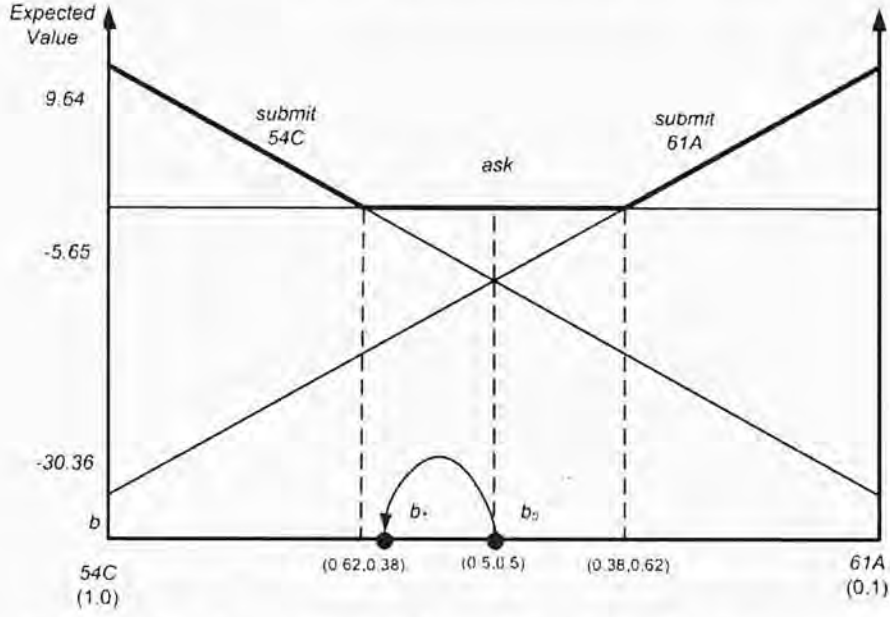


Figure 3.4: The optimal policy of the NaiveBusInfo system.

3.2 represented as black dots in Figure 3.4 for example. We can see that the belief state is transited from b_0 to b_1 after belief updating, but b_1 still stays in the central region. Therefore, according to the policy, the system should take the *ask* action in the next step.

Belief updating and policy optimization by maximizing the accumulated rewards constitute the POMDP framework and form the basis for building a POMDP system. The remaining of this Chapter introduces the SDS-POMDP model for dialog modeling in [56] and an efficient algorithm for policy optimization CSPBVI in [42]. Based on the theories, a POMDP-based dialog system in bus

information domain is constructed.

3.2 The SDS-POMDP Model

The SDS-POMDP model is an extension of the regular POMDP framework by integrating with the characteristics of SDSs [18]. In SDS-POMDP, the POMDP action a represents the dialog system action. The POMDP state s has been factorized into three components, *i.e.*, the true user action a_u , the true user goal g , and dialog history h . Therefore the belief state becomes the joint distribution over a_u , g , and h :

$$b(s) = b(g, a_u, h). \quad (3.6)$$

The POMDP observation o stands for the recognition results from ASR, and is expressed by the recognition hypothesis \hat{a}_u and the recognition confidence score c , *i.e.*, $o = (\hat{a}_u, c)$. The observation model in POMDP becomes:

$$P(o'|s', a) = P(\hat{a}'_u, c|g', a'_u, h', a). \quad (3.7)$$

Since the observation is primarily corrupted by the ASR component, the observation is assumed to depend only on the current user's action:

$$P(o'|s', a) = P(o'|a'_u) = P(\hat{a}'_u, c|a'_u). \quad (3.8)$$

The factor graph of SDS-POMDP is shown in Figure 3.5. Based on the inference diagram of SDS-POMDP and Bayesian rules, the state transition probability could be factored as below,

$$P(s'|s, a) = P(g'|g, h, a_u, a)P(a'_u|g', g, h, a_u, a)P(h'|a'_u, g', g, h, a_u, a)^1. \quad (3.9)$$

¹ x' in this thesis refers to the state. (*e.g.*, user goal or observation) in the next time step.

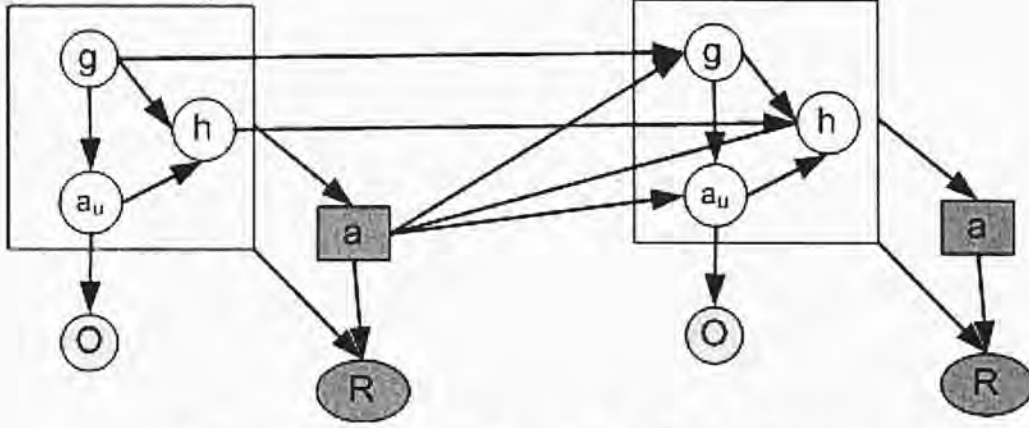


Figure 3.5: The factor graph of SDS-POMDP

The first term in Equation 3.9 is called *user goal model*. The SDS-POMDP model assumes the user's goal at each time-step only depends on the previous user's goal, the dialog history, and the system's action:

$$P(g'|g, h, a_u, a) = P(g'|g, h, a). \quad (3.10)$$

The second term in Equation 3.9 is *user action model*. It is assumed in SDS-POMDP that the user's action at each time-step only depends on the current user's goal, and the previous system's action:

$$P(a'_u|g', g, h, a_u, a) = P(a'_u|g', a). \quad (3.11)$$

The last one is called *dialog history model*, and it is simplified as below:

$$P(h'|a'_u, g', g, h, a_u, a) = P(h'|a'_u, g', h, a). \quad (3.12)$$

Now the simplified transition model becomes:

$$P(s'|s, a) = P(g'|g, h, a)P(a'_u|g', a)P(h'|a'_u, g', h, a). \quad (3.13)$$

The models of the three factors and the observation model constitute the basic set of parameters of SDS-POMDP. As Williams [18] pointed out, the

user goal model and the user action model can be typically estimated from an annotated dialog corpus. The history model can either be estimated from a dialog corpus or handcrafted. The observation model can be derived using language model, *etc.*

Based on equations 3.7, 3.8, 3.10, 3.12, and 3.1, the belief monitoring process in SDS-POMDP model is updated as below:

$$b_a^o(s') = b_a^{\hat{a}_u, c}(g', a'_u, h') \propto P(\hat{a}'_u, c | a'_u) \cdot P(a'_u | g', a) \cdot \sum_g P(g' | g, h, a) \sum_h P(h' | a'_u, g', h, a) \sum_{a_u} b(g, h, a_u). \quad (3.14)$$

Based on the definition of SDS-POMDP, the operation of the factored model is the same as that of the standard POMDP which has been illustrated in Section 3.1.1.

3.3 Composite Summary Point-based Value Iteration (CSPBVI)

The SDS-POMDP model successfully addresses the problem of uncertainties due to speech recognition for dialog modeling [18]. However, when the number of user goals is increasing, the number of POMDP states is exponentially increasing, which results in intractability in model optimization. Hence, this model can only be limited to a small number of user goals, which cannot satisfy the demands of real users. In order to build a real-world system, the CSPBVI is proposed by Williams *et al.* to solve the scalability problem [42].

The key idea of CSPBVI is to construct an SDS-POMDP for each slot in a summary space and separately learn an optimally simply policy for each slot-level POMDP. A simply heuristic is designed to combine these simply policies into a composite one in the master space [42]. This algorithm consists of four stages, *i.e.*, construction of POMDPs, sampling belief points in summary space, optimization of the simply POMDP for each slot, and execution based

on the action choosing heuristics.

Suppose the slot-filling dialog system to be built has W slots. A master SDS-POMDP and W summary POMDPs is constructed. Master POMDP is for the whole system and is parameterized by a original set of user goals and user actions, while summary POMDP is for only one slot and is characterized by a simplified set of user goals (*i.e.*, best and rest). Detailed description of master and summary POMDPs is illustrated in Table 3.1. We can observe that the states in summary POMDP have exponentially decreased to 6 from $|G|^W$. This states reduction saves much computation for policy optimization.

Components	Master POMDP	Summary POMDP for slot w
State	$s = (g, a_u, h)$	$\hat{s}_w = (\hat{g}, \hat{h})$
User goal	$G = \{g g = (g^1, g^2, \dots, g^w)\}$	$\hat{G} = \{best, rest\}$
History	$H = \{h h = (h^1, h^2, \dots, h^w)\}$ $h^w \in \{n, u, c\}$	$\hat{H} = \{n, u, c\}$
System action	$a_s = predicate[w](g^w)$	$\hat{a}_s = predicate(\hat{g})$

Table 3.1: Description of the components for the master and summary POMDPs. The *predicate* refers to the illocutionary force of the action. \hat{x} stands for the component in the summary space. n means the slot value has not been stated, u means the slot value has not been confirmed, and c means the slot value has been confirmed.

Different from sampling the belief points in the master space in the classical PBVI algorithm, CSPBVI samples N belief points in the summary space for each slot respectively, *i.e.*, $\{b_{w,n}\}_{n=1}^{n=N}$, where $w = 1, 2, \dots, W$. During the sampling, the slot's dynamics and the corresponding rewards are also recorded. After, the optimization is performed for each slot by using the recorded dynamics and rewards, and the learned policy for slot w is represented as $\{b_{w,n}, \hat{a}_{w,n}, r_{w,n}\}_{n=1}^{n=N}$, where $\hat{a}_{w,n}$ and $r_{w,n}$ are the system action and its immediate reward for the n th point respectively. According to the W policies, given a belief point b in the master space, the execution process can be summarized as follows:

1. Transform the belief point b into the summary one \hat{b}_w for slot w ,
2. Find the sampled point $\hat{b}_{w,n}$ nearest to \hat{b}_w for slot w ,
3. Get the system action \hat{a}_w in the summary space based on the simple optimal policy for slot w , where $\hat{a}_w = \hat{a}_{w,n}$,
4. Jump to step 1 until all the slots have been navigated,
5. Choose an appropriate system action \hat{a}_x among these slot actions $\{\hat{a}_1, \dots, \hat{a}_w\}$ based on the handcrafted action choosing heuristics,
6. Transform \hat{a}_x into a that is in the master space.

3.4 Application of SDS-POMDP Model: The BusInfo System

In this section, a statistical dialog system BusInfo is developed, where the task is to provide bus timetable information in Pittsburgh, Pennsylvania. The BusInfo system will track the distribution over dialog states for each slot in real time, display the best hypothesis and propose an appropriate action for each slot based on the distribution.

3.4.1 System Description

The BusInfo system is built based on the CSPBVI algorithm described in the previous section. In each dialog, the user should provide the origin, destination, and bus number. After, the system will search for the corresponding schedule results. Therefore, there are 3 slots in the system and the summarizations of the slots are shown in Table 3.2.

Slot	Size	Example values
Origin	13	Airport, Downtown, Greenfield, South Oakland
Destination	13	Waterfront, Shadyside, Bloomfield, Lawrenceville
Bus route	4	28X, 54C, 61A, 64A

Table 3.2: Summarizations of the slots information in the BusInfo system

Components	Mater POMDP	Summary POMDP
User action	$ A_u = 32$	$ \hat{A}_u = null$
User goal	$ G = 676$	$ \hat{G} = 2$
History	$ H = 27$	$ \hat{H} = 3$
State	$ S = 584064$	$ \hat{S} = 6$
System action	$ A = 709$	$ \hat{A} = 4$

Table 3.3: Summarizations of the components in the BusInfo system

In the BusInfo system, there are 32 user actions; 30 actions² are about stating the values of the three slots (see Table 3.2), like “want to take the 28X bus” which states the bus route “28X”, and the remaining two are “yes” and “no” for system’s confirmation. The observation in the system consists of a user action and the corresponding confidence score. Hence, there are 584064 ($13 * 13 * 4 * 3^3 * 32$) states in total in the master POMDP, and 6 in the summary space. The predicates of the systems actions are “ask”, “submit”, and “confirm”. When translating the predicates into system actions according to the definition in Table 3.1, there are 709 system actions in the master POMDP and 4 in each summary POMDP. The summarizations of the components in the system are exhibited in Table 3.3.

The parameters of the *user action model* $P(a'_u|g', a)$ are estimated from 200 dialogs collected through the Let’s Go! dialog system [4] by using frequency counting, shown in Table 3.4. The Let’s Go! dialog system is developed by CMU and will be detailed in Chapter 4. The *reward model* gives the most reward for submitting the correct value for the slot. It penalizes the most for submitting the incorrect value for the slot. Other actions, such as the “ask”

²Each of the 30 slot values for the three slots (see Table 3.2) corresponds to one user action.

$a = \text{predicate}[w_s](\text{value}_s)$	$a_u = (w_u, \text{value}_u)$	$P(a'_u g', a)$ $g' = (g^1, \dots, g^w)$
$\text{predicate} = \text{Ask}$	$w_s = w_u \ \& \ \text{value}_u = g^{w_u}$	0.95
$\text{predicate} = \text{Ask}$	$w_s = w_u \ \& \ \text{value}_u = \text{null}$	0.05
$\text{predicate} = \text{Ask}$	$w_s \neq w_u \ \& \ \text{value}_u = g^{w_u}$	0.17
$\text{predicate} = \text{Ask}$	$w_s \neq w_u \ \& \ \text{value}_u = \text{null}$	0.83
$\text{predicate} = \text{Confirm}$ & $\text{value}_s = g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = \text{yes}$	0.75
$\text{predicate} = \text{Confirm}$ & $\text{value}_s = g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = g^{w_u}$	0.22
$\text{predicate} = \text{Confirm}$ & $\text{value}_s = g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = \text{null}$	0.03
$\text{predicate} = \text{Confirm}$ & $\text{value}_s \neq g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = \text{no}$	0.78
$\text{predicate} = \text{Confirm}$ & $\text{value}_s \neq g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = g^{w_u}$	0.19
$\text{predicate} = \text{Confirm}$ & $\text{value}_s \neq g^{w_s}$	$w_s = w_u \ \& \ \text{value}_u = \text{null}$	0.03
$\text{predicate} = \text{Confirm}$	$w_s \neq w_u \ \& \ \text{value}_u = g^{w_u}$	0.32
$\text{predicate} = \text{Confirm}$	$w_s \neq w_u \ \& \ \text{value}_u = \text{null}$	0.68

Table 3.4: Summarizations of user action model in the BusInfo system

and “confirm”, are penalized much lighter than the wrong submission. Details about reward model are shown in Table. The *user goal model* keeps user goal consistent over the course of the dialog, *i.e.*, $P(g'|g, h, a) = 1$ if $g' = g$. The *discount factor* γ (see Equation 3.3) is selected as 0.99.

The BusInfo system uses the text input component as the input module, *i.e.*, users communicate with the system by typing text rather than speak. However, the input texts cannot reflect speech recognition errors which are one of the main uncertainties in spoken dialog system, and cannot verify that the POMDP-based system could work robust to the recognition uncertainties. Hence, we create a simulated ASR to generate some recognition errors for the inputs on the concept-level. The simulated recognition errors are limited to concept substitution for each slot and are generated with probability P_{err} which is set as 0.3 in BusInfo system. For example, the input concept “Waterfront”

$a = \text{predicate}[w_s](\text{value}_s)$	$h = (h^1, \dots, h^w)$	$r(a, h, g)$ $g = (g^1, \dots, g^w)$
$\text{predicate} = \text{Ask}$	$h^{w_s=n}$	-1.5
$\text{predicate} = \text{Ask}$	$h^{w_s=u}$	-3
$\text{predicate} = \text{Ask}$	$h^{w_s=c}$	-4.5
$\text{predicate} = \text{Confirm}$	$h^{w_s=n}$	-4
$\text{predicate} = \text{Confirm}$	$h^{w_s=u}$	-1
$\text{predicate} = \text{Confirm}$	$h^{w_s=c}$	-2.5
$\text{predicate} = \text{Submit}$	$\text{value}_s = g^{w_s}$	20
$\text{predicate} = \text{Submit}$	$\text{value}_s \neq g^{w_s}$	-20

Table 3.5: Summarizations of reward model in the BusInfo system

for the Origin slot can be substituted with “Greenfield” or “54C”. Each simulated recognition concept is associated with a recognition confidence score. As studied in the past research work, the confidence score follows an exponential distribution [57] which has also been applied in Williams’ work [42]. Similarly, the confidence score for correct concept in BusInfo system is sampled from the density function $p_h(c)$, while that for the incorrect concept is sampled from $p_h(1 - c)$:

$$p_h(c) = \frac{he^{hc}}{e^h - 1}, \quad (3.15)$$

where c is confidence score. Therefore, the observation model can be defined as follows:

$$P(o|a_u) = P(\hat{a}_u, c|a_u) = \begin{cases} (1 - P_{err}) * p_h(c) & \text{if } \hat{a}_u = a_u \\ \frac{P_{err}}{|A_u|} * p_h(1 - c) & \text{otherwise.} \end{cases} \quad (3.16)$$

Figure 3.6 shows the architecture of BusInfo system. The dialog policy of each slot is learned by CSPBVI optimization beforehand. When the system works, it first updates the belief state based on the observation (the simulated recognition results) and makes a decision on the action according to the policy. The whole system, including both optimization and belief monitoring, is implemented in Visual Studio with C#. Some simple grammar rules are designed to identify

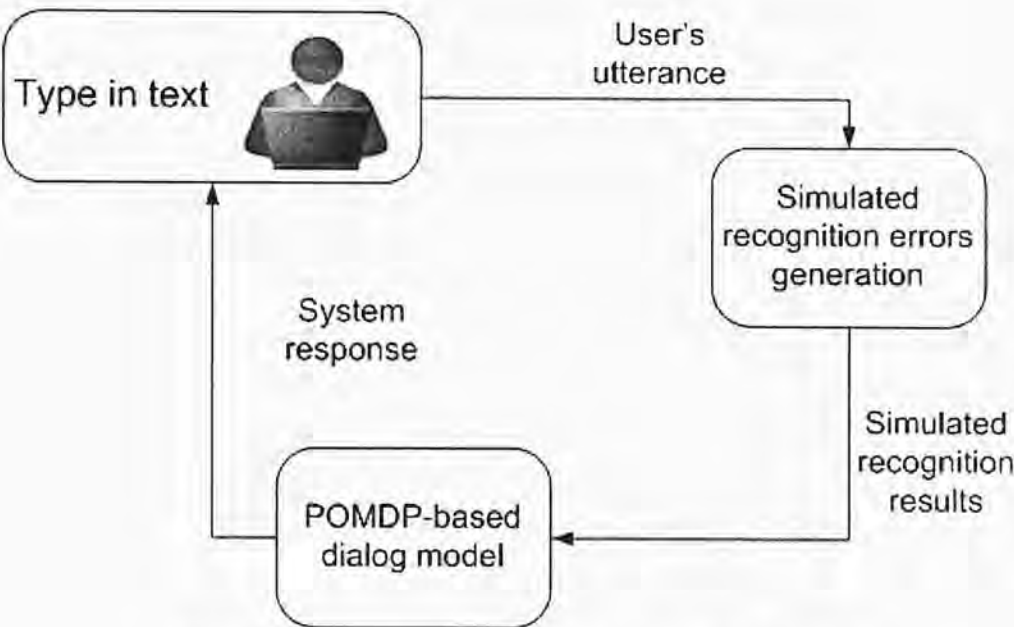


Figure 3.6: The architecture of BusInfo system

the slot value. Due to the limited definition of the *user action model* in CSPBVI, the system can recognize only one slot value from the user’s inputs at a time. A set of simple templates are also created to generate system utterance from the abstract representation of system action.

3.4.2 Demonstration Description

Figure 3.7 shows the screen shot of the BusInfo interface. This demonstration consists of four components. The lower left component is the interactive part where users can interact with the system by typing text in the *User* text box and clicking the *Run* button. In a real system, this should interface with transcribed speech by an automatic speech recognizer. Below the *User* text box are the simulated recognition results. The system gives prompts in the right above window, such as “Which bus do you want to take?” in this case. The *Start Over* button helps you to restart a new dialog.

The lower right component is the POMDP dialog model which demonstrates the current status. The *Summary Belief States $b(best)$* column shows the belief

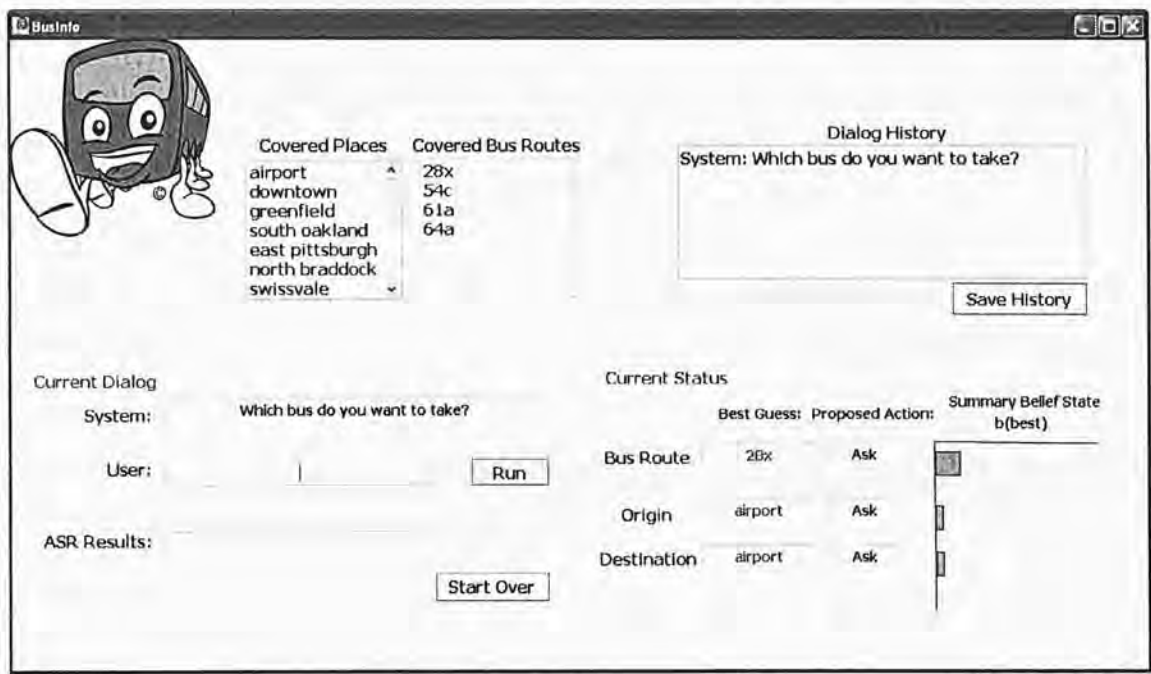


Figure 3.7: Screen shot of the BusInfo system showing the beginning of the dialog. It consists of four parts, *i.e.*, interaction part, dialog status part, dialog history part, and covered information part. In the beginning of the dialog, the initial belief over the user goals for each slot is uniform.

of the most likely user goal for each slot. The Best Guess column shows the most likely user goal for each slot. The middle column exhibits the action for each slot which is proposed based on the optimal policy. In the beginning of a dialog, the proposed actions for all the slots are “Ask”.

The upper left part shows the covered bus routes and places of the system. Users can refer to them when using the system. The dialog discourse will be recorded in the dialog history window during the user-system interaction. It can also be saved into a .txt file by clicking the *Save History* button.

An example dialog with the BusInfo system is shown in Table 3.7. In each turn, the upper part shows the user-system interaction, while the lower part shows the dialog status, including the belief of the most likely user goal in each slot, the proposed actions, and the most likely user goals. In this dialog, the user is trying to search for the route of 54C from *Greenfield* to *North Braddock*. In the beginning of the dialog, each slot nominates the “Ask” action and the

initial belief over the user goals for each slot is uniform. Therefore, as Table 3.7 shows, the initial belief of the four possible *Bus Routes* is 0.25, and the probability of the thirteen possible *Origins* or *Destinations* is 0.077.

This dialog illustrates how the POMDP-based dialog system plans in the environment with recognition errors. The system asks user for information slot by slot. In turn 1, the user wants to take the “54C” bus route but “54C” is mis-recognized for “Waterfront”. Since “Waterfront” is not a Bus Route, the initial belief keeps unchanged. Next, the system keeps asking for bus route, since it didn’t get any information about it previously. This time, “54C” is recognized correctly with a recognition score 0.756, which leads to a large belief mass shifting to “54C”. The Bus Route slot proposes the “Confirm” action, and the other slots propose the “Ask” action. In turn 3, “54C” receives more belief mass shift resulting from an explicit confirmation, causing the Bus Route slot nominates the “Submit” action. In turn 4, the user’s response “Greenfield” is correctly recognized with a high recognition score 0.99, so the Origin slot proposes the “Submit” action directly. In turn 5, the user’s response “North Braddock” is recognized with a lower confidence score 0.38, resulting in only a slight shift of belief mass to “North Braddock”. The Destination slot keeps asking in the next turn where “North Braddock” is recognized with a higher score. The accumulation of belief mass shifts toward “North Braddock” leads to the “Submit” action in the Destination slot. After all the slots propose “Submit”, the system decides to submit the user’s request to database for search.




























Five users in our lab are invited to interact with BusInfo to schedule bus trips, and 30 dialogs are collected. Table 3.6 shows an example of the BusInfo dialogs. Based on these dialogs, we apply our evaluation model to assess the performance of the BusInfo system as will be described in Chapter 5.

S:	<i>Which bus do you want to take?</i>
U:	<i>54c</i>
S:	<i>Where are you leaving from?</i>
U:	<i>east pittsburgh</i>
S:	<i>From east pittsburgh, right?</i>
U:	<i>Yes</i>
S:	<i>Where are you going to?</i>
U:	<i>bloomfield</i>
S:	<i>Ok! Your route 54c from east pittsburgh to bloomfield is searched!</i>

Table 3.6: An example of the BusInfo dialogs.

3.5 Chapter Summary

This chapter describes the formulation and planning of POMDP, and provides an introduction to a statistical model of SDS-POMDP for dialog modeling which factorizes the POMDP state into three distinct components, *i.e.*, the user's goal, the user's action, and the dialog history. Then it describes an efficient optimization algorithm of scaling POMDP-based SDSs of a real-world size, *i.e.*, Composite summary point based value iteration (CSPBVI). By applying CSPBVI, a statistical dialog system in bus information domain called BusInfo is created based on SDS-POMDP model. During the human-system interaction, the system tracks the distribution over the dialog states, and a real-time visualization is provided which provides the best hypothesis, belief of the most likely state and proposed action for each slot. Through interacting with the system, users can easily understand how the statistical technique improves the robustness to recognition errors in SDS. This system will be evaluated by the proposed paradigm for SDS evaluation in Chapter 4 and 5.

Initial status	<p>Current Status</p> <table border="1"> <thead> <tr> <th></th><th>Best Guess:</th><th>Proposed Action:</th><th>Summary Belief State b(best)</th></tr> </thead> <tbody> <tr> <td>Bus Route</td><td>28x</td><td>Ask</td><td></td></tr> <tr> <td>Origin</td><td>airport</td><td>Ask</td><td></td></tr> <tr> <td>Destination</td><td>airport</td><td>Ask</td><td></td></tr> </tbody> </table>		Best Guess:	Proposed Action:	Summary Belief State b(best)	Bus Route	28x	Ask		Origin	airport	Ask		Destination	airport	Ask	
	Best Guess:	Proposed Action:	Summary Belief State b(best)														
Bus Route	28x	Ask															
Origin	airport	Ask															
Destination	airport	Ask															
Turn 1	<p>Current Dialog</p> <p>System: Which bus do you want to take?</p> <p>User: <input type="text" value="want take 54c"/> <input type="button" value="Run"/></p> <p>ASR Results: <input type="text" value="waterfront [0.145882045475968]"/> <input type="button" value="Start Over"/></p> <p>Current Status</p> <table border="1"> <thead> <tr> <th></th><th>Best Guess:</th><th>Proposed Action:</th><th>Summary Belief State b(best)</th></tr> </thead> <tbody> <tr> <td>Bus Route</td><td>28x</td><td>Ask</td><td></td></tr> <tr> <td>Origin</td><td>airport</td><td>Ask</td><td></td></tr> <tr> <td>Destination</td><td>airport</td><td>Ask</td><td></td></tr> </tbody> </table>		Best Guess:	Proposed Action:	Summary Belief State b(best)	Bus Route	28x	Ask		Origin	airport	Ask		Destination	airport	Ask	
	Best Guess:	Proposed Action:	Summary Belief State b(best)														
Bus Route	28x	Ask															
Origin	airport	Ask															
Destination	airport	Ask															
Turn 2	<p>Current Dialog</p> <p>System: Which bus do you want to take?</p> <p>User: <input type="text" value="take 54c"/> <input type="button" value="Run"/></p> <p>ASR Results: <input type="text" value="54c [0.755989706166458]"/> <input type="button" value="Start Over"/></p> <p>Current Status</p> <table border="1"> <thead> <tr> <th></th><th>Best Guess:</th><th>Proposed Action:</th><th>Summary Belief State b(best)</th></tr> </thead> <tbody> <tr> <td>Bus Route</td><td>54c</td><td>Confirm</td><td></td></tr> <tr> <td>Origin</td><td>airport</td><td>Ask</td><td></td></tr> <tr> <td>Destination</td><td>airport</td><td>Ask</td><td></td></tr> </tbody> </table>		Best Guess:	Proposed Action:	Summary Belief State b(best)	Bus Route	54c	Confirm		Origin	airport	Ask		Destination	airport	Ask	
	Best Guess:	Proposed Action:	Summary Belief State b(best)														
Bus Route	54c	Confirm															
Origin	airport	Ask															
Destination	airport	Ask															

Turn 3	Current Dialog			
	System:		Take bus 54c,right?	
	User:		yes	Run
	ASR Results:		yes[1]	
			Start Over	
	Current Status			
		Best Guess:	Proposed Action:	Summary Belief State b(best)
	Bus Route	54c	Submit	<div></div>
	Origin	airport	Ask	<div></div>
	Destination	airport	Ask	<div></div>

Turn 4	Current Dialog			
	System:		Where are you leaving from?	
	User:		from greenfield	Run
	ASR Results:		greenfield [0.999667054988897]	
			Start Over	
	Current Status			
		Best Guess:	Proposed Action:	Summary Belief State b(best)
	Bus Route	54c	Submit	<div></div>
	Origin	greenfield	Submit	<div></div>
	Destination	airport	Ask	<div></div>

Turn 5	<div>Current Dialog</div> <div>System: <div>Where are you going to?</div></div> <div>User: <div>going to north braddock</div> <div>Run</div></div> <div>ASR Results: <div>north braddock [0.379864836093451]</div> <div>Start Over</div></div> <div>Current Status</div> <table><thead><tr><th></th><th>Best Guess:</th><th>Proposed Action:</th><th>Summary Belief State b(best)</th></tr></thead><tbody><tr><td>Bus Route</td><td>54c</td><td>Submit</td><td><div></div></td></tr><tr><td>Origin</td><td>greenfield</td><td>Submit</td><td><div></div></td></tr><tr><td>Destination</td><td>north</td><td>Ask</td><td><div></div></td></tr></tbody></table>		Best Guess:	Proposed Action:	Summary Belief State b(best)	Bus Route	54c	Submit	<div></div>	Origin	greenfield	Submit	<div></div>	Destination	north	Ask	<div></div>
	Best Guess:	Proposed Action:	Summary Belief State b(best)														
Bus Route	54c	Submit	<div></div>														
Origin	greenfield	Submit	<div></div>														
Destination	north	Ask	<div></div>														
Turn 6	<div>Current Dialog</div> <div>System: <div>Where are you going to?</div></div> <div>User: <div>north braddock</div> <div>Run</div></div> <div>ASR Results: <div>north braddock [0.751283866314491]</div> <div>Start Over</div></div> <div>Current Status</div> <table><thead><tr><th></th><th>Best Guess:</th><th>Proposed Action:</th><th>Summary Belief State b(best)</th></tr></thead><tbody><tr><td>Bus Route</td><td>54c</td><td>Submit</td><td><div></div></td></tr><tr><td>Origin</td><td>greenfield</td><td>Submit</td><td><div></div></td></tr><tr><td>Destination</td><td>north</td><td>Submit</td><td><div></div></td></tr></tbody></table>		Best Guess:	Proposed Action:	Summary Belief State b(best)	Bus Route	54c	Submit	<div></div>	Origin	greenfield	Submit	<div></div>	Destination	north	Submit	<div></div>
	Best Guess:	Proposed Action:	Summary Belief State b(best)														
Bus Route	54c	Submit	<div></div>														
Origin	greenfield	Submit	<div></div>														
Destination	north	Submit	<div></div>														
Turn 7	<div>Current Dialog</div> <div>System: <div>Ok! Your route 54c from greenfield to north braddock is searched!</div></div>																

Table 3.7: An example dialog with the BusInfo system. In each turn, the upper part shows the user-system interaction, while the lower part shows the dialog status, including the belief of the most likely user goal in each slot, the proposed actions, and the most likely user goals.

Chapter 4

Collecting User Judgments on Spoken Dialogs with Crowdsourcing

As introduced in Chapter 2, the subjective user judgments on SDSs are considered the most important criterion for SDSs performance. However, user judgments are usually measured by inviting subjects to fill out a questionnaire after they interact with an SDS, which is a costly, tedious and time-consuming process. This chapter presents an initial attempt at the use of crowdsourcing for collection of user judgments on SDSs [58]. This is implemented on Amazon Mechanical Turk (MTurk), where a Requester can design a human intelligence task (HIT) to be performed by a large number of Workers efficiently and cost-effectively. We describe a design methodology for two types of HITs – the first targets at fast rating of a large number of dialogs regarding some dimensions of the SDSs performance and the second aims to assess the reliability of Workers on MTurk through the variability in ratings across different Workers. In addition, a set of approval rules are developed. They are necessary to exclude random submissions with nonsensical ratings, which will affect the overall quality of the ratings obtained.

This chapter first describes the experimental dialog corpus and a simple

S:	<i>Welcome to the CMU Let's Go bus information system</i>
S:	<i>What can I do for you?</i>
U:	<i>When is the next 64A from Forbes and Murray to East Liberty?</i>
S:	<i>The 64A. Did I get that right?</i>
U:	<i>Yes.</i>
S:	<i>Leaving from Forbes and Murray. Correct?</i>
U:	<i>Yes.</i>
S:	<i>Going to East Liberty. Am I right?</i>
U:	<i>Yes</i>
S:	<i>I think you want the next bus.</i>
U:	<i>Yes</i>
S:	<i>Hold on. I'll look that up.</i>
S:	<i>The next 64A leaves Murray Avenue at Forbes at 10:32p.m. and arrives at Highland Avenue at Penn Avenue at 10:55p.m.</i>

Table 4.1: An example of the Let's Go! dialogs.

dialog classification algorithm in terms of task completion, then introduces the methodology of collecting user judgments using crowdsourcing. After, collected results are exhibited, together with analysis of the results. Finally, the reliability of MTurk for outsourcing evaluation of SDSs is examined by measuring the agreement between MTurk Workers and experts.

4.1 Dialog Corpus and Automatic Dialog Classification

The corpus used in this study contains about 50,000 dialogs drawn from the log files collected through the Let's Go! dialog system developed by Carnegie Mellon University [4]. This system provides bus schedule information for Pittsburgh residents when the customer service hotline is not staffed. Input utterances from users are transcribed automatically by automatic speech recognition (ASR). Table 4.1 shows an example of the Let's Go! dialogs.

We survey the dialog logs and identify several cue phrases that enable us

to develop a set of heuristics to automatically classify the dialog logs into five categories in terms of task success. The dialog logs are tracked from the end towards the beginning with a set of ordered rules:

1. Dialogs with no more than 3 turns are classified as the “TooShort” category.
2. Dialogs with the cue phrases, “Start a new query”, “New query”, etc., in the user turns are classified as the “MultiTask” category.
3. Dialogs with the cue phrase “...result¹ ...” in the system turns are classified as the “TaskComplete” category.
4. Dialogs with the cue phrase “I’m sorry...” in the system turns are classified as the “OutOfScope” category.
5. If none of the above rules apply, the dialog is classified as the “TaskIncomplete” category.

The example dialogs in the five categories are shown in Figure 4.1. In TooShort dialogs, the user usually hung off with only one user utterance which may result from that the user doesn’t really want to use the system. Therefore, such dialogs are meaningless and are separated from the TaskIncomplete ones. The distribution of the classification results is shown in Figure 4.2. About 35% of the dialogs are in the TaskComplete category, while the remaining categories share comparable percentages. As will be described later, this rough classification is used to determine the proportion of the different types of dialogs that are uploaded to MTurk for crowdsourcing. The rough classification also provides a means by which we can verify whether the human raters on the MTurk platform have devoted efforts in rating our dialogs, as opposed to providing random (and unreliable) inputs. We will elaborate on these points in the following sections.

¹This cue phrase is observed in the grammar tags of the Let’s Go! System’s dialog logs.

Case 1: TaskComplete S: Welcome to the CMU Let's Go bus information system. S: What can I do for you? U: When is the next 64A from Forbes and Murray to East Liberty? S: The 64A. Did I get that right? U: Yes. S: Leaving from Forbes and Murray. Correct? U: Yes. S: Going to East Liberty. Am I right? U: Yes S: I think you want the next bus. U: Yes S: Hold on. I'll look that up. S: The next 64A leaves Murray Avenue at Forbes at 10:32pm and arrives at Highland Avenue at Penn Avenue at 10:55 pm.
Case 2: TooShort S: Welcome to the CMU Let's Go bus information system. S: What can I do for you? U: OAKLAND .ANY. .FIND. .SOUTH SIDE .WANT.
Case 3: MultiTask S: There is a 28X leaving THE AIRPORT ... U: NEW QUERY S: Okay, let's start from the beginning.
Case 4: OutofScope S: I'm sorry, I don't know any routes...
Case 5: TaskIncomplete S: Leave at 8 a.m. Is this correct? U: YES S: Hold on, I'll look that up.

Figure 4.1: Samples of the Let's Go! dialogs in the categories of automatic classification.

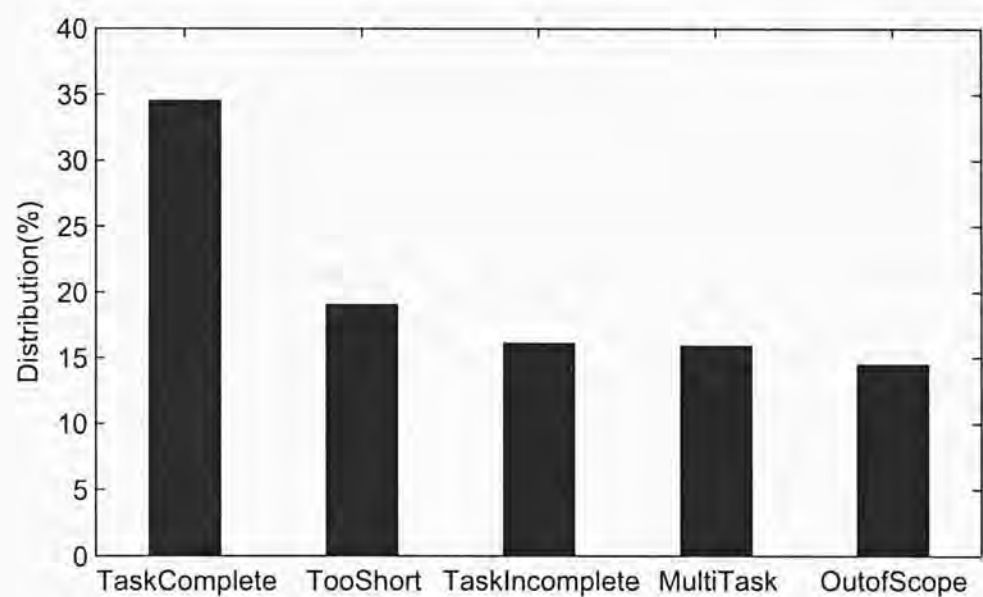


Figure 4.2: Distribution of the five dialog categories based on the automatic, heuristics-based classification.

4.2 User Judgments Collection with Crowdsourcing

The objective of collecting user judgments with crowdsourcing is to get a large number of gathered dialogs evaluated by numerous people (hence more statistically representative) in an efficient and cost-effective manner, which is difficult to be realized using traditional methods. The MTurk platform organizes work in the form of human intelligence tasks (HITs). An HIT is designed by the “Requester” (i.e., our research team) and is completed by many “Workers” (i.e., anyone who is interested in the task) over the Internet. Requesters must specify the payment per HIT, the number of unique Workers to work on each HIT, the maximum time a Worker has to work on a single task, *etc.* Requesters can set qualification criteria for Workers to meet to work on the HITs. Qualifications may include approval rate, Worker’s location, *etc.* In this HIT, the Worker’s approval rate should be higher than or equal to

98%. The Workers' inputs to the HITs will undergo an "approval process" by the Requester. Approved inputs will result in automatic payment from the Requester to the Worker via the MTurk platform.

4.2.1 HITs on Dialog Evaluation

This type of HITs are designed to outsource the assessment of the SDS to MTurk Workers. The assessment focuses on selected dimensions of performance regarding the SDS, based on a large number of selected dialogs from the logs. To achieve this goal, we have authored a set of questions that constitute the HIT in Table 4.2.

As shown in Table 4.2, we include the explanation of the aim for each question, but this is not shown to the MTurk Workers. These questions cover *the user's confidence, the perceived task completion, the expected behavior, the overall performance and the categorization of task success*. We choose these questions because our data includes only the textual transcription of a dialog, which implies that they do not involve interaction with real users. Hence, we cannot design HIT and collect user feedback through crowdsourcing. The question, "Do you think the output speech is of good quality?" is an example to illustrates our point. In particular, for Question 5, the initial set of three answer options led to much disagreement among Workers. Many also sent us comments about the lack of a clear definition of task completion versus task incompleteness. Consequently, we revised to include seven answer options (see Table 4.3) based on the ITU Recommendation [2]. We have purposely designed the questions in such a way that they can cross-validate each other (Q2 and Q5 both aim to assess task completion), which will be used for approval of ratings from MTurk later.

Each HIT contains the text transcription of one dialog and the questionnaire in Table 4.2 for assessment by the Workers, who are paid USD \$0.05 for

Q1	Do you think you understand from the dialog what the user wanted?
Opt	1) No clue 2) A little bit 3) Somewhat 4) Mostly 5) Entirely
Aim	<i>elicit the Worker's confidence in his/her ratings.</i>
Q2	Do you think the system is successful in providing the information that the user wanted?
Opt	1) Entirely unsuccessful 2) Mostly unsuccessful 3) Half successful/unsuccessful 4) Mostly successful 5) Entirely successful
Aim	<i>elicit the Worker's perception of whether the dialog has fulfilled the informational goal of the user.</i>
Q3	Does the system work the way you expect it?
Opt	1) Not at all 2) Barely 3) Somewhat 4) Almost 5) Completely
Aim	<i>elicit the Worker's impression of whether the dialog flow suits general expectations.</i>
Q4	Overall, do you think that this is a good system?
Opt	1) Very poor 2) Poor 3) Fair 4) Good 5) Very good
Aim	<i>elicit the Worker's overall impression of the SDS.</i>
Q5	What category do you think the dialog belongs to?
Opt	1) Task is incomplete 2) Out of scope 3) Task is complete
Aim	<i>elicit the Worker's impression of whether the dialog reflects task completion.</i>

Table 4.2: Questions constituting the HIT on Dialog Evaluation (Q: Question, Opt: Options). The questionnaire covers *the user's confidence, the perceived task completion, the expected behavior, the overall performance and the categorization of task success.*

TS:S	Succeeded (task for which solutions exist)
TS:Cs	Succeeded with constraint relaxation by system
TS:Cu	Succeeded with constraint relaxation by the user
TS:CsCu	Succeeded with constraint relaxation both from the system and from the user
TS:SN	Succeeded in spotting that no solution exists
TS:F _s	Failed because of the system behavior, due to system inadequacies
TS:F _u	Failed because of the user behavior, due to non-cooperative user behavior

Table 4.3: Definitions of different levels of task success, based on the ITU Recommendation [2]. These definitions serve as the options to question 5 in the questionnaire.

each task completed. We have uploaded 11,000 dialogs in total, including samples from the three major dialog categories and in proportions that follow the percentages obtained from the automatic classification, i.e., TaskComplete (55%), TaskIncomplete (27%), OutofScope (18%). TooShort and MultiTask dialogs are excluded from the HIT. The former is easily detectable as unsuccessful. The latter can be easily segmented into mono-task dialogs, which can then follow the three-way categorization (TaskComplete / TaskIncomplete / OutofScope) directly.

4.2.2 HITs on Inter-rater Agreement

This type of HITs are the extensions of those in Section 4.2.1 and are designed to assess the reliability of MTurk Workers through inter-rater agreement across different raters. Each HIT includes the text transcriptions of 30 selected Let’s Go! dialogs drawn from log files (10 dialogs from the categories of TaskComplete, TaskIncomplete and OutofScope respectively). Each dialog is associated with the questionnaire in Table 4.2. Workers are paid USD \$1.5 for each task completed. Altogether, we have 3 groups of Workers (each with 16 individuals)

rating two sets of dialogs (each with 30). Groups 1 and 2 evaluate the first set of dialogs, while Group 3 evaluate the second set. In this way, we can assess whether the inter-rater agreement varies across different raters and different dialogs.

4.2.3 Approval of Ratings

It is important to verify the quality of inputs from a large number of MTurk Workers. Since the quality of the ratings directly impacts the credibility of the SDS evaluation, some basic rules have to be set to ensure the Workers are devoting efforts and to guarantee the reliability of ratings, in addition to the qualification requirement preset for the Workers. We have developed the approval mechanism, as follows:

- R1. We reject HITs for which the working time is less than 15 seconds, since we feel that careful (and thus high quality) ratings cannot be completed within such a short period.
- R2. If an MTurk Worker completes a large number of HITs (e.g., over 20) but provides identical answers for all of them, his/her work will be rejected.
- R3. Approval requires consistency between the answers to related questions (Q2 and Q5). Consistency is based on four main heuristics:
 - Answers to Q2 being “Entirely successful” or “Mostly successful” can go with answers to Q5 being TS:S, TS:CS, TS:Cu, or TS:CsCu.
 - Answers to Q2 being “Entirely unsuccessful” or “Mostly unsuccessful” can go with answers to Q5 being TS:F_s or TS:F_u.
 - The answer to Q2 being “Half unsuccessful / successful” can go with any answer in Question 5.
 - The answer to Q5 being TS:SN can go with any answer to Q2.

R4. Approval requires consistency between the answers to Q5 and the automatic classification of the dialogs (see Section 4.1). In particular, the heuristics are:

- TaskComplete can match with TS:S, TS:Cs, TS:Cu and TS:CsCu.
- TaskIncomplete can match with TS:Fs and TS:Fu.
- OutofScope can match with TS:SN.

R5. If these above heuristics are not satisfied, the dialog will be checked carefully. Random (incorrect) ratings are rejected. However, we have approved some ambiguous cases, as they will be explained in Section 4.3.2.

4.3 Collected Results and Analysis

4.3.1 Approval Rates and Comments from Mturk Workers

11,000 HITs are rated by around 700 online Workers in 45 days. Three persons in our team complete the verification of the rated HITs and approve 8,394 of them. The total expenditure paid to the Workers is USD \$350. Approval rates for each dialog category, i.e., TaskComplete, TaskIncomplete and OutofScope, are shown in Figure 4.3 respectively. OutofScope is the highest because some Workers consider a task to be successful if they think that the absence of the information is due to the database but not the ability of the system. Others consider such cases as failures since the system does not provide the requested information for the users. We approve either decision from the Workers.

Rejected dialogs led to some controversies. Some apologized for their errors and others complained about the rejections. We received feedbacks from the MTurk Workers concerned, many of which are useful to help enhance our understanding of SDS evaluations. Here we list some typical comments

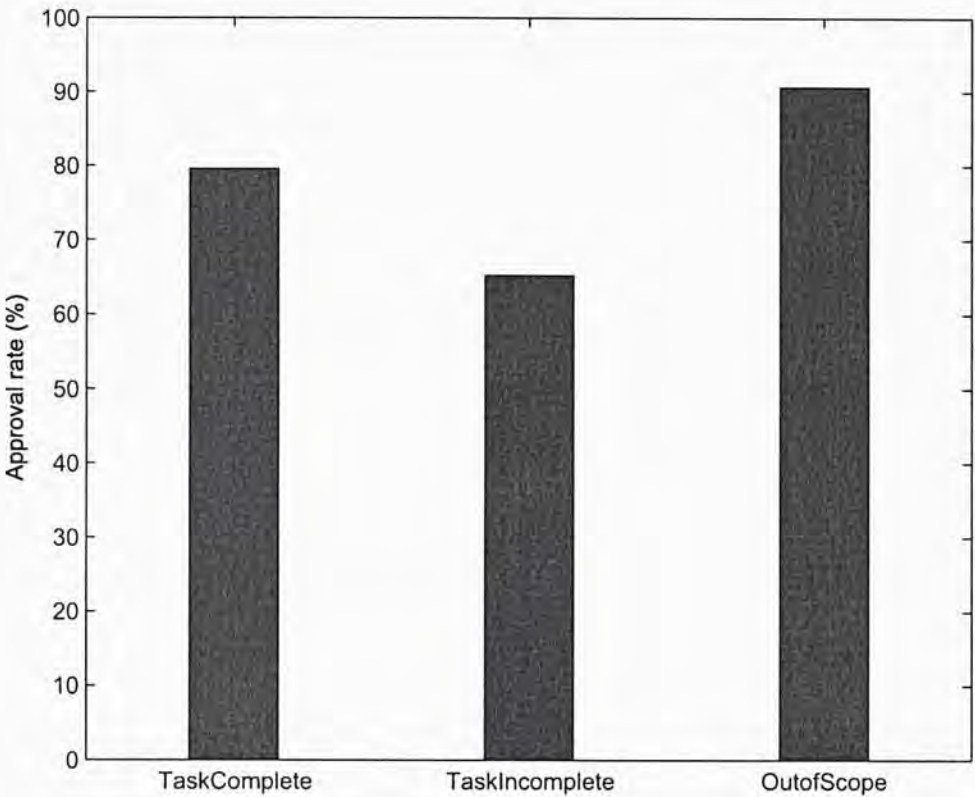


Figure 4.3: Approval rates for each dialog category, i.e., TaskComplete, TaskIncomplete and OutofScope. OutofScope has the highest approval rate.

associated with their implications as follows.

- The system does not provide exact information that the user wanted although it provides some related results. (*Retrieval result from database is a vital aspect of SDS performance.*)
- The understanding ability of system is very important on the user's first try, so good understanding may lead workers to choose task success even if the system does not provide any information to the user. (*Good language understanding ability plays an important role in improving user satisfaction.*)
- The system succeeds in providing a message based on the user's initial inputs but fails to follow up with the user's updated information. (*Timely updating the dialog history impacts users' perception on SDS performance positively and greatly.*)

4.3.2 Consistency between Automatic Dialog Classification and Manual Ratings

To assess the quality of ratings from MTurk, we investigate the consistency between automatic dialog classification (see Section 4.1) and the manual ratings from MTurk Workers (with respect to Questions 2 and 5) based on the approved HITs of about 8,000 dialogs.

According to the approval rules, the HITs whose ratings of Q5 are not consistent with the automatic classification will be rejected (see R4 in Section 4.2.3). However, as mentioned in Section 4.2.3, we still approve some ambiguous dialogs for which it is difficult to determine their success in task completion. The manual ratings of Q5 for the ambiguous dialogs do not agree with automatic classification. Table 4.4 shows an example, where the dialog is terminated midway. Some Workers regard the dialog as successful (in Question 5), because

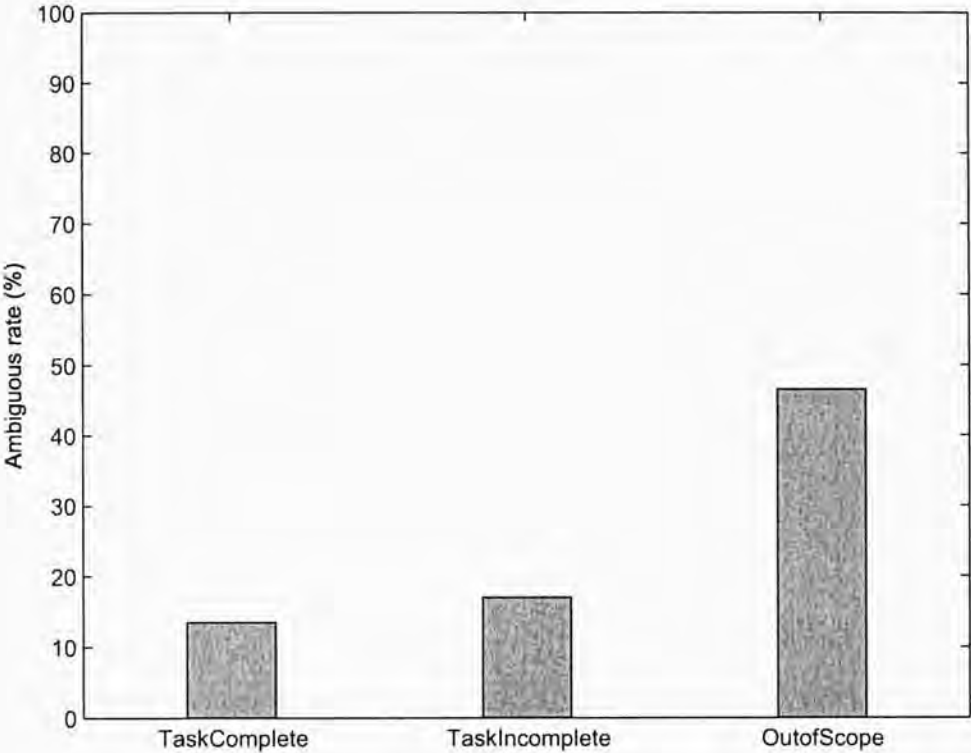


Figure 4.4: Percentages of ambiguous dialogs over the total approved ones in each category of TaskComplete, TaskIncomplete and OutofScope. A higher percentage of ambiguous dialogs are approved for the OutOfScope category.

System:	<i>Welcome to the CMU Let's Go bus information system.</i>
System:	<i>To get help at any time, just say Help or press zero.</i>
System:	<i>What can I do for you?</i>
User:	<i>TWENTY. .EIGHT. .AT. .IN. .DOWNTOWN .FIFTY. AIRPORT.</i>
System:	<i>Leave from AIRPORT. Is this correct?</i>
User:	<i>FORM DOWNTOW</i>
System:	<i>Leave from DOWNTOWN. Is this correct?</i>
User:	<i>YES</i>
System:	<i>Where are you going to?</i>
User:	<i>TO AIRPORT</i>
System:	<i>You want to go to AIRPORT. Is this correct?</i>
User:	<i>YES</i>
System:	<i>What time are you leaving?</i>
User:	<i>EIGHT A M</i>
System:	<i>leave at 8 a. m. Is this correct?</i>
User:	<i>YES</i>
System:	<i>Alright</i>
System:	<i>Hold on. I'll look that up.</i>

Table 4.4: An ambiguous dialog from TaskIncomplete category.

they feel that the system would have completed the task had the user not hung up. Nevertheless, since the dialog does not have the necessary cue phrase (i.e. “..result..”) in the log, it is automatically classified as TaskIncomplete. Percentages of ambiguous dialogs over the total approved ones in each category of TaskComplete, TaskIncomplete and OutofScope are shown in Figure 4.4 respectively. Note that a higher percentage of ambiguous dialogs are approved for the OutOfScope category, mainly due to MTurk Workers’ diversified understandings of such kind of dialogs, as we have discussed in Section 4.3.1.

Moreover, Figure 4.5 plots the mean answer scores of the two questions for approved dialogs in each of the three categories, where a higher score maps to a higher level of success, namely:

- Answer scores to Q2 range from 0 for “Entirely unsuccessful” to 1 for

“Entirely successful”.

- Answer scores to Q5 range from 0 for “TS:Fu” to 1 for “TS:S”.

The scores to Q2 and Q5 have been normalized in the same range from 0 to 1.

Generally, despite some ambiguous dialogs are approved, we still observe reasonable agreement, i.e., the dialogs automatically classified as TaskComplete receive high scores from the Workers, those automatically classified as TaskIncomplete receive low scores, and those in OutofScope category receive neutral scores. Such consistency verifies the reliability of the approved ratings from MTurk to some extent.

4.3.3 Inter-rater Agreement Among Workers

As mentioned earlier, the second type of HITs (see Section 4.2.2) are designed to assess the level of inter-rater agreement (ITA) among the MTurk Workers. We adopt Cohen’s weighted kappa measure which is often applied to ordinal categories,

$$Kappa = \frac{\sum_{i=1}^c \sum_{j=1}^c w_{ij} (n_{ij}/N - n_{i.}n_{.j}/N^2)}{1 - \sum_{i=1}^c \sum_{j=1}^c w_{ij} n_{i.}n_{.j}/N^2}, \quad (4.1)$$

where c is the number of categories (i.e., answer options for each question here, $c = 5$ for Q1-Q4 and $c = 7$ for Q5), $w_{ij} = 1 - \frac{(i-j)^2}{(1-c)^2}$, n_{ij} is the element in the observed matrix, $n_{i.} = \sum_j n_{ij}$ and $n_{.j} = \sum_i n_{ij}$. Details can be found in [59]. A higher kappa value indicates a higher inter-rater agreement.

Recall that we have three groups of Workers rating two sets of dialogs. These ratings are accepted directly and do not undergo the approval process. For any pair of Workers in each group, we compute the weighted kappa value for each question. We then compute the mean weighted kappa value for each question over the entire group. Results are shown in Figure 4.6. Despite the fact that groups 1 and 2 evaluated the same dialog set, while group 3

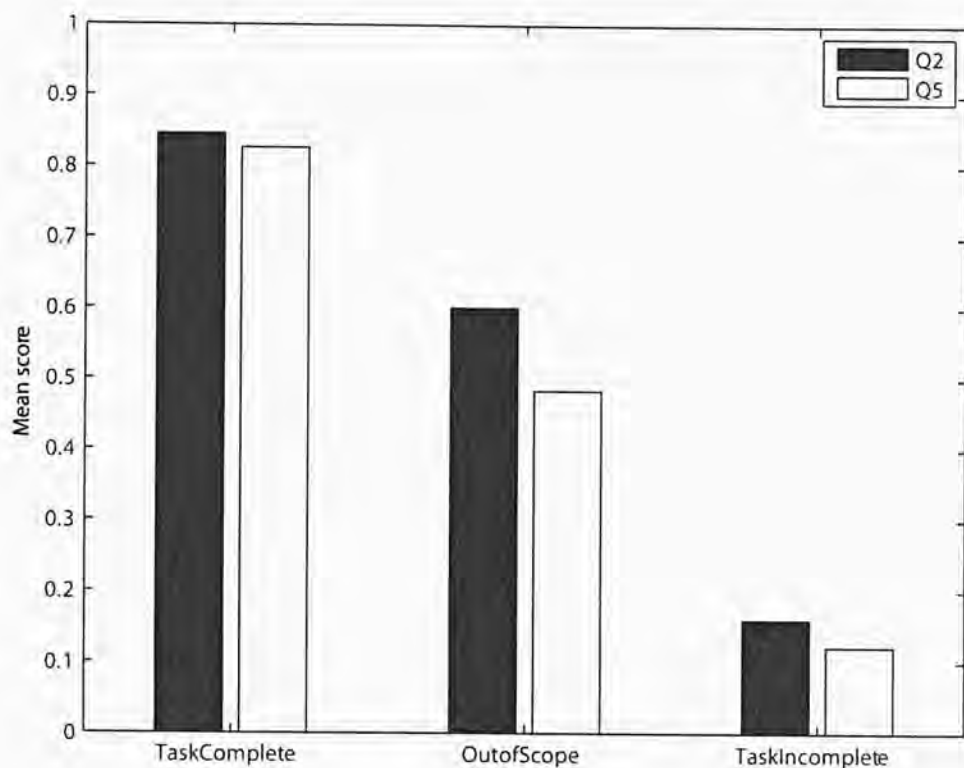


Figure 4.5: The normalized mean scores of Q2 and Q5 for approved ratings in each category. A higher score maps to a higher level of task success.

evaluated a different dialog set, the three kappa plots remain close, which illustrates that the inter-rater agreement for each question remains stable across different raters and different dialogs. In particular, Q5 (categorization of task success) achieves mean weighted kappa values above 0.6 and Q2 (perceived task completion) achieves reasonable values above 0.4, which is indicative of a moderate level of agreement [60]. Q2 and Q5 are about task success which can gain “official” or somehow objective ratings from reliable raters, so the moderate and stable agreement partially shows the reliability of MTurk Workers and provides support for the utilization of MTurk as a judgment collection platform. On the other hand, Q3 (expected behavior) and Q4 (overall impression on system performance) have low values below 0.3, which

is indicative of a lack of agreement. This suggests that evaluation based on overall user satisfaction may be quite subjective. The low agreement in user satisfaction may lead to the low prediction accuracy for the evaluation model, which has been analyzed in [27].

Figure 4.6 also shows an interesting observation that for Q2 and Q5, the relative value gap between group 1 and 2 is larger than that between group 2 and 3. Intuitively, the values of inter-rater agreement in group 1 and 2 should be closer, since Workers in the two groups evaluate the same set of dialogs. We investigate the annotations of Workers in each group. Results show that three Workers in group 1 are obviously unreliable. For example, they rate the TaskComplete dialog in Table 4.5 as TaskIncomplete which contradicts with the rates of most group members, while there are no such inconsistent inputs in the other two groups. These inconsistent annotations from the few unreliable Workers lead to the lower inter-rater agreement in group 1. Despite this difference, the level of inter-rater agreement over the two groups is still the same. If we exclude the ratings from the three unreliable Workers, the inter-rater agreement of Q2 will increase from 0.444 to 0.542, and that of Q5 will increase from 0.621 to 0.698.

We conceive of several factors that cause low inter-rater agreement in overall user satisfaction. First, different users may emphasize different aspects of system performance, ranging from the system's intelligence, task completion, dialog efficiency, etc. Second, raters with different levels of domain knowledge may have different expectations of the system. It may not be meaningful to compute an overall average score of user satisfaction across a diversity of users. Instead, we may consider the following possibilities:

- Evaluate SDSs along different dimensions individually.
- Evaluate SDSs in terms of different types of user queries, targeting at different system functionalities.

S: Welcome to the CMU Let's Go bus information system.
S: To get help at any time, just say Help or press zero.
S: What can I do for you?
U: I'D LIKE TO GO FROM FORBES AND BIGELOW TO
SQUIRREL HILL
S: Leaving from FORBES AND BIGELOW. Did I get that right?
U: YES
S: Right. Going to SQUIRREL HILL. Did I get that right?
U: YES
S: Okay. When do you want to travel?
U: NEXT AVAILABLE
S: When would you like to travel?
U: RIGHT NOW
S: You want the next bus. Is this correct?
U: YES
S: Just a minute. I'll look that up.
S: The next 61A leaves FORBES AVENUE AT BIGELOW
SCHENLEY DRIVE at 10 40 p.m. and arrives at FORBES
AVENUE AT MURRAY at 10 49 p.m.

Table 4.5: A TaskComplete dialog from the dialog set in the HIT on inter-rater agreement.

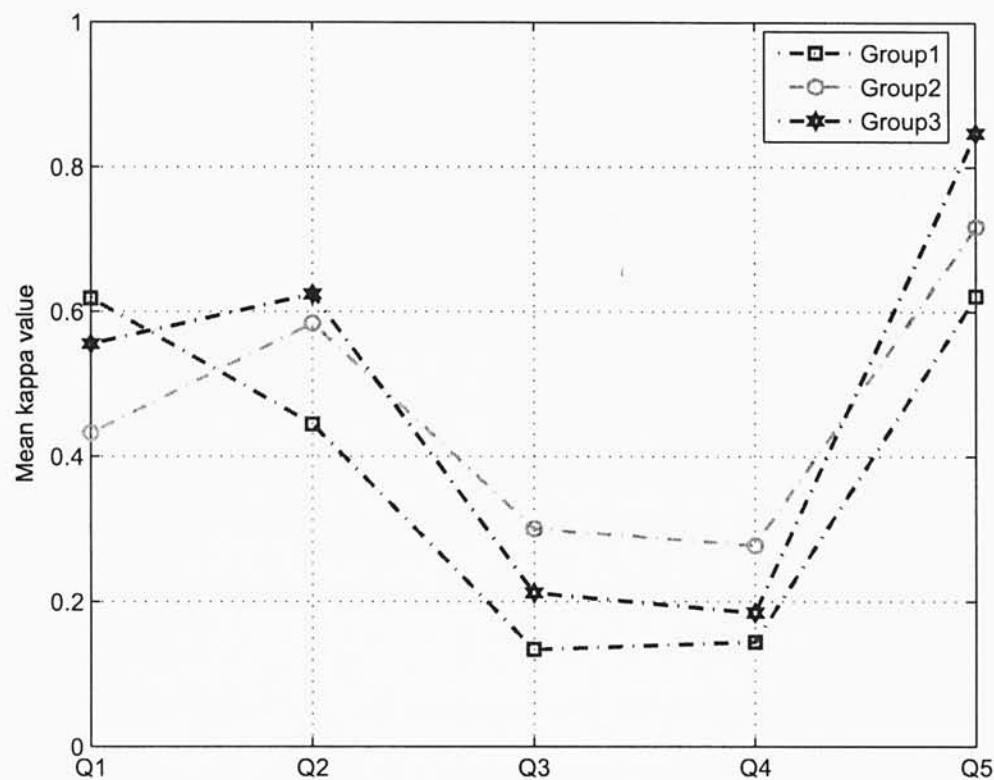


Figure 4.6: Mean values of weighted Kappa of five questions for 3 groups. The Kappa values are stable across different raters and different dialogs. Q2 and Q5 which can gain “official” ratings from reliable raters have high levels of inter-rater agreement.

- Evaluate SDSs based on different user groups with different levels of domain knowledge.

4.4 Comparing Experts to Non-experts

Section 4.3 has demonstrated the reliability of the SDS evaluation from MTurk Workers by comparing the crowdsourced annotations to automatic classification and analyzing the inter-rater agreement among Workers. Nonetheless, there remains a key issue not resolved: whether the SDS evaluation by MTurk

Workers is as reliable as that by experts². In this section, we will investigate the level of agreement between the non-expert and expert annotations through two case studies.

4.4.1 Inter-rater Agreement on the Let's Go! System

The second type of HIT in Section 4.2.2 contains a set of 30 dialogs from the Let's Go! dialog corpus, of which each 10 dialogs are from the categories of TaskComplete, TaskIncomplete and OutofScope. We have 4 experts evaluate the set of dialogs and compare their results with those of one group of 16 Workers who evaluate the same set of dialogs in Section 4.2.2. The weighted Kappa in Equation 4.1 is still adopted to measure the inter-rater agreement (IRA) [59].

For any pair of experts, we first compute the IRA value for each question. We then calculate the mean expert-expert ($E-E$) IRA for each question over all the pairs. Given an expert E , the IRA value of any pair of E and Workers is calculated, and then the E -Workers ($E-W$) IRA is obtained by averaging the IRA values over all the E -Worker pairs. Intuitively, reliable Workers are expected to have a high level of agreement with experts.

Table 4.6 shows $E-W$ and $E-E$ IRA values for each question. There's a high level of agreement between experts and Workers for question 2 and 5 (the kappa values are around 0.7) which are about classification of task completion and represent relatively objective ratings [60], and the $E-W$ IRA values approach those of $E-E$ IRA. This agreement indicates that Workers perform well on these measures and the crowdsourced data could be considered reliable. On the other hand, similar to the results of IRA among Workers in Section 4.3.3, Q3 (expected behavior) and Q4 (overall impression on system performance) have lower IRA values for both $E-W$ and $E-E$, which is indicative

²The expert in this chapter refers to the researchers in the area of spoken dialog systems.

of lower agreement. Although agreement is slightly higher among the experts, it remains lower overall and suggests that evaluation based on overall user satisfaction may be quite subjective.

	E_1-W	E_2-W	E_3-W	E_4-W	$E-E$
Q1	0.5625	0.6107	0.5508	0.5525	0.6937
Q2	0.6744	0.7037	0.7160	0.7434	0.8709
Q3	0.2838	0.4368	0.2441	0.4214	0.5288
Q4	0.3686	0.4295	0.3188	0.4416	0.6240
Q5	0.7984	0.8242	0.8209	0.81	0.9435

Table 4.6: Expert-Workers and Expert-Expert agreement. There’s a high level of agreement between experts and Workers for question 2 and 5 which are about classification of task completion, while the other questions about the overall user satisfaction have a lower agreement.

4.4.2 Consistency Between Expert and Non-expert Annotations on SDC Systems

The 2010 Spoken Dialog Challenge (SDC) is organized by Dialog Research Center at CMU. The aim of this challenge is to realistically compare different spoken dialog system in the same dialog task and make the collected dialogs from the systems available for the development of evaluation techniques.

In the 2010 SDC, there are four participating systems (the three competitors and the CMU reference system). All these systems provide Pittsburgh bus schedule information. Four dialog corpora was collected through the SDC controlled test which was conducted by asking subjects to call all the four systems. The organizers manually transcribed all the dialogs for each system and manually labeled task success for each dialog. A dialog is labeled as successful, if one piece of acceptable information is provided [61]. We publish the text transcriptions of all the dialogs on MTurk using the HIT template in Section 4.2.1. Each dialog is evaluated by three Workers. Task completion of a dialog is determined by the answers to Q5 with the use of majority vote. For

example, if two or more Workers choose one of the options of TS:S, TS:SCs, TS:SCu and TS:SCsCu, the dialog will be tagged as success.

We compare the task completion labels from Workers with those from CMU experts. Several measures are defined for comparison. Given a dialog corpus $D = \{d_1, d_2, \dots, d_N\}$, the labels of the corpus from MTurk Workers are denoted as $L^W = \{l_1^W, l_2^W, \dots, l_N^W\}$, and labels from CMU experts are denoted as $L^E = \{l_1^E, l_2^E, \dots, l_N^E\}$, where $l_i^W, l_i^E \in \{Success, Failure\}$. The first measure called success rate (SR) is defined in the following way,

$$SR = \frac{\sum_{i=1}^N I(l_i = Success)}{N}, \quad (4.2)$$

where $I(\cdot)$ is:

$$I(\omega) = \begin{cases} 1 & \text{if } \omega = \text{true} \\ 0 & \text{otherwise.} \end{cases} \quad (4.3)$$

The the second measure of consistency rate (CR) is defined as below,

$$CR = \frac{\sum_{i=1}^N I(l_i^W = l_i^E)}{N}. \quad (4.4)$$

The third measure is consistent success rate (CSR):

$$CSR = \frac{\sum_{i=1}^N I((l_i^W = Success) \& (l_i^E = Success))}{\sum_{i=1}^N I((l_i^W = Success) \vee (l_i^E = Success))}. \quad (4.5)$$

Comparison results are shown in Table 4.7. The SR per system by MTurk Workers is quite close to that by the CMU experts. The CR s are all around 70%, even above 80% for system 3 and 4. The CSR s (around 80%) indicate that the “Success” annotations from experts and non-experts have a large overlap. Table 4.7 demonstrates Worker judgments are generally consistent with expert judgments. This consistency also provides support for the utilization of MTurk as a platform for collecting user judgments on dialogs.

	Sys1	Sys2	Sys3	Sys4
Total dialogs	91	61	75	83
SR (MTurk)	56.1%	36.1%	86.7%	78.3%
SR (CMU)	64.8%	37.3%	89.3%	74.7%
CR	70.3%	73.7%	85.3%	86.7%
CSR	88.1%	80%	85.9%	88.5%

Table 4.7: Comparison of labels from CMU experts and Workers.

Next, we investigate the cases where the Workers differ from the experts. We find that experts care more about the accuracy of the bus information that the system provides when they label the dialogs. In some dialogs, the systems provide the bus information that cannot exactly meet the user’s request, for example, wrong bus arrival time though correct bus route. The experts regard such dialogs as failure, while Workers often consider them as success. On the other hand, Workers seem to focus on the user’s final intent. In some cases where the user changes his mind and triggers a new query after the system provides acceptable information, the experts tag them as success, while Workers often label them as failure since they feel that the system does not accomplish the user’s final goal. Although the classification of task completion is more objective than other ratings (like user satisfaction), there are still some differences of opinion which may cause inconsistency.

4.5 Chapter Summary

This chapter presents our initial attempt at the use of crowdsourcing for collection of user judgments on spoken dialog systems through MTurk. We describe a design methodology for two types of HITs - the first targets fast collection of ratings of a large number of dialogs efficiently and the second aims to assess the reliability of the evaluation from MTurk through inter-rater agreement among Workers. A set of approval rules are developed to take care of the quality of ratings from MTurk.

Compared with the traditional method of inviting subjects to fill out a questionnaire after interaction, the results we achieved show that the crowdsourcing method is more efficient, flexible and inexpensive, and could access more statistically representative population. At the same time, the quality of ratings can also be controlled. Reliable ratings for 8,394 dialogs rated by around 700 online Workers are approved. Approval rates for each dialog category, i.e., TaskComplete, TaskIncomplete and OutofScope, are 79.59%, 65.23% and 90.65% respectively. Reasonable consistency between the manual MTurk ratings and the automatically classified dialogs in terms of task success is an indicator of the reliability of the approved ratings from MTurk. The moderate level of inter-rater agreement among Workers for ratings in task completion partially verifies the reliability of MTurk Workers.

This chapter has also provided support for the use of MTurk for crowdsourcing evaluation of SDSs by investigating the agreement between Workers and experts through two case studies. Experimental results showed a high level inter-rater agreement (around 0.7) between experts and MTurk Workers in terms of task completion when we compared their annotations on the Let's Go! dialog corpus. There was also a high consistency between expert and non-expert labels of task success for the dialogs from the four SDC systems. The collected corpus of ratings on the Let's Go! system from MTurk will be used to develop a quantitative SDS evaluation framework in Chapter 5.

Chapter 5

Collaborative Filtering for Performance Prediction

The previous chapter introduces a method of collecting user judgments on SDSs with crowdsourcing. Based on the collected corpus of user judgments, developing accurate models to automatically predict user satisfaction about the overall quality of an SDS is highly desirable for SDS evaluation. In the original PARADISE framework, a linear regression model is trained using measures drawn from rated dialogs as predictors with user satisfaction as the target. In this chapter, we extend PARADISE by introducing a collaborative filtering (CF) model for user satisfaction prediction [62]. This prediction model is drawn from the idea of CF in recommendation systems, which uses information from near neighbors of an unrated dialog to predict its user satisfaction. Then the basic CF model is extended by considering user judgments with respect of *user style* and *system quality*.

This chapter first gives a brief introduction about CF and details our approach. Then, experimental results are presented and analyzed. After, the generalizability of this evaluation model is verified on multiple SDSs within the bus information domain. Finally, the CF model is applied to evaluate the performance of the BusInfo system built in Chapter 3.

5.1 Item-Based Collaborative Filtering

Collaborative filtering (CF) uses a database of users' preferences for items to predict the utility of a certain item for a particular user. Item-based techniques are one main category among CF's implementations. These methods search for items most similar to the target one in a data set which has been rated by users. Prediction of the target item is then computed based on similar ones. Item-based CF is computationally efficient and can guarantee recommendation quality [63].

Suppose that the k most similar items of the target i are selected for the active user u , and their ratings by u are denoted as $\{r_{u,j}\}_{j=1}^k$. A typical way to predict the rating $P_{u,i}$ of the target item i for the user u is to compute the weighted sum of ratings on the k similar items,

$$P_{u,i} = \frac{\sum_{j \in \{k \text{ similar items}\}} s_{i,j} * r_{u,j}}{\sum_{j \in \{k \text{ similar items}\}} s_{i,j}}, \quad (5.1)$$

where the weights $\{s_{i,j}\}_{j=1}^k$ are similarities between i and the k items. For some more elaborate algorithms for item-based CF we refer readers to [31].

While our proposed algorithms are inspired by item-based CF, we want to highlight some differences between the SDS evaluation problem and CF. First, items in our problem are more consistent than those in recommendation systems—they are all dialogs. This unique characteristic allows us to represent the items by some common features (see Section 5.3), and the similarity between two dialogs is hence computed from their feature vectors. Secondly, the dialogs similar to the target may be rated by different users, so we do not intend to predict the rating of the target dialog for a particular user u , but rather for a general population of users.

5.2 CF Model for User Satisfaction Prediction

5.2.1 ICFM for User Satisfaction Prediction

We detail our item-based CF model (ICFM) for user satisfaction prediction in the following. Let $D = \{(d_i, r_i)\}_{i=1}^N$ be a large dialog corpus where each dialog d_i is rated as r_i . As pointed out in the previous section, we represent each dialog d_i with a feature vector \mathbf{f}_i which has been normalized to its z score, and the similarity between two dialogs d_i and d_j is measured as the cosine similarity of their feature vectors,

$$s_{i,j} \doteq s(d_i, d_j) = \frac{\mathbf{f}_i^T \mathbf{f}_j}{|\mathbf{f}_i| * |\mathbf{f}_j|}. \quad (5.2)$$

To save computation time, we cluster dialog corpus using k -means in advance. Let $C = \{C_i\}_{i=1}^M$ be the clusters created from D such that $\cap_i C_i = \phi$ & $\cup_i C_i = D$. Therefore, the retrieval process of k similar dialogs for the target dialog d relates to its assignment to a cluster C^* ,

$$C^* = \arg \max_{C_i} s(d, c_i), \quad (5.3)$$

where c_i is the centroid of C_i .

Sarwar et al. pointed out that two items with high similarity may be distant in Euclidean distance [64], therefore they proposed to map the known rating $r_{u,j}$ in Eq. 5.1 to $g(r_{u,j})$. When $g(\cdot)$ is a linear mapping, it reduces to the linear regression problem. Hence we use linear regression trained on the selected cluster C^* to predict the rating for the target dialog d , rather than use the weighted sum (see Eq. 5.1). Note that since we have partitioned the dialog corpus into M clusters, the linear regression can be trained on each cluster beforehand.

With such modifications, ICFM is formulated as below,

1. Extract feature vector \mathbf{f}_i for each dialog $d_i \in D$.

2. Use k -means to create dialog clusters C for the dialog corpus D based on the feature representations \mathbf{f} and the similarity measure in Eq. 5.2.
3. Build linear regression models $L = \{L_i | r = L_i(\mathbf{f})\}_{i=1}^M$ for the created clusters, which means model L_i is trained from dialogs in cluster C_i .
4. Given an unseen dialog d (unevaluated dialog here), we first extract a feature vector \mathbf{f}_d and then assign d into cluster C^* with Eq. 5.3.
5. Use L^* which is trained on C^* to predict user satisfaction for d .

5.2.2 Extended ICFM for User Satisfaction Prediction

By considering the characteristics of dialogs which record interactions between users and an SDS, we find that the features extracted (see Section 5.3) can be separated into *user-related* and *system-related* types. For example, #Barge In (overall number of user's barge in attempts) reflects the characteristics of user behavior and can be classified as a user-related feature, while #System Question (overall number of system's questions in the dialog) is a system-related feature. The intuition for this separation is that judgement rating for a dialog can be influenced by two types of features, i.e., user style and system quality. On one hand, users with different user styles may have different tastes for the dialog, which can result in different evaluations for the same dialog. On the other hand, a high-quality dialog coming from the system is more likely to get a high rating statistically. Ratings determined by the user style can be obtained from user-related features and those due to the system quality can be drawn from system-related ones. Hence, we can predict judgement ratings based on the two types of features *separately*, rather than on the basis of the entire feature set. Based on this idea, we extend ICFM to EICFM as follows,

1. Create system-related clusters C^s for dialog corpus D based on system-related features \mathbf{f}^s .

2. Create user-related clusters C^u for D based on user-related features \mathbf{f}^u .
3. Build linear regression models L^s and L^u for C^s and C^u respectively.
4. Given an unseen dialog d , choose C^{s*} and C^{u*} which are most similar to d with respect to system-related features \mathbf{f}^s and user-related ones \mathbf{f}^u , respectively.
5. Use regression model L^{s*} to predict system-related judgement r^s for d , and use model L^{u*} to predict user-related judgement r^u .
6. The final rating r is obtained by linearly combining the two kinds of ratings, $r = r^u * w + r^s * (1 - w)$, where w is a weight varying from 0 to 1. This weight is determined by a validation set in our experiments.

Compared with ICFM, EICFM can have a better balance between user judgments from user style and system quality. As will be seen, experiments demonstrate that this extension distinctly improves the evaluation performance.

5.3 Extraction of Interaction Features

In this section, we describe how to extract the feature vector \mathbf{f}_i for each dialog d_i in dialog corpus D .

The dialog corpus D used in this chapter is consisting of 5000¹ Let's Go! dialogs on which the user judgments have been collected with crowdsourcing through MTurk in Chapter 4. As will be seen in Section 5.4, our data is used in a 10-fold cross validation style in the first experiment. In the second step, we divide the corpus into training and test set, containing 4,000 and 1,000 dialogs respectively.

According to ITU Recommendation [2], we extract some interaction features, whose meanings are illustrated in Table 5.1, from the log files for each dialog.

¹We only use the dialogs rated with the revised questionnaire according to the ITU recommendation in Chapter 4.

These features are chosen since they can be automatically extracted from the log files which recording the interactions. The features of **#Help Requests** and **#User Questions** are obtained based on some cue phrases such as help, what, where, *etc.* The features of **#System Turns**, **#User Turns**, **AveRecogScore**, **#Barge In** and **#Help Requests** were used in the original version of the PARADISE model [24], while **#DTMF** is specific to the Let's Go! system since it provides touch tone functionality to users. Therefore, each dialog is represented by a vector \mathbf{f}_i concatenating all the features.

Feature	Definition
#System Turns	Overall number of system turns
#User Turns	Overall number of user turns
WPUT	Average number of words per user turn
AveUserSpeakRate	Average speaking rate of user's
AveRecogScore	Average recognition score
#Barge In	Overall number of user's barge in attempts
#Help Requests	Overall number of user's help requests
#User Questions	Overall number of user's questions
#System Questions	Overall number of system's questions
#DTMF	Overall number of touch tone uses

Table 5.1: Features automatically extracted from log files. Each dialog is represented by a vector \mathbf{f}_i concatenating all the features.

Among these features, **#System Turns**, **AveRecogScore** and **#System Questions** are classified as *system-related* ones for they are mostly influenced by the characteristics of the system, while the others are determined by user behavior and are hence *user-related*. All features use z-norm scores in the following experiments.

5.4 Experimental Results and Analysis

We conduct two experiments to investigate how ICFM and EICFM can improve user satisfaction prediction. **Experiment I** compares R^2 in predicting user satisfaction for ICFM, EICFM and the linear regression model (LRM) using 10-fold cross validation. **Experiment II** is to compare the mean values of true ratings and predictions of the test data over the number of system turns (**#System Turns**), because the LRM results show that this feature takes on the largest weight.

For convenience, we set the number of user-related clusters C^u to be equal to that of system-related clusters C^s in EICFM in all the experiments. The weighting w is set to 0.1 empirically through the use of a validation set. We use R^2 to measure the prediction accuracy in **Experiment I**,

$$R^2 = 1 - \frac{\sum_{i=1}^n (r_i - \hat{r}_i)^2}{\sum_{i=1}^n (r_i - \bar{r})^2}, \quad (5.4)$$

where r_i is the ground truth rating, \hat{r}_i is the predicted rating from a prediction model, and \bar{r} is the mean of $\{r_i\}_{i=1}^n$. The higher R^2 is, the higher the prediction accuracy is.

5.4.1 Prediction of User Satisfaction

In **Experiment I**, we use 10-fold cross validation on the data corpus (5,000 rated dialogs, see Section 5.3) to measure R^2 in predicting user satisfaction of test data for ICFM, EICFM and LRM. Recall that Q3 and Q4 in the questionnaire (see Table 4.2) cover the user's expectation and overall impression, therefore the responses to Q3 and Q4 for each dialog are averaged to yield a single user satisfaction rating ranging from 1 to 5 as the output of the prediction model, while the input is the 10-dimensional feature vector introduced in Section 5.3.

Figure 5.1 shows R^2 of predicting user satisfaction changing with the cluster

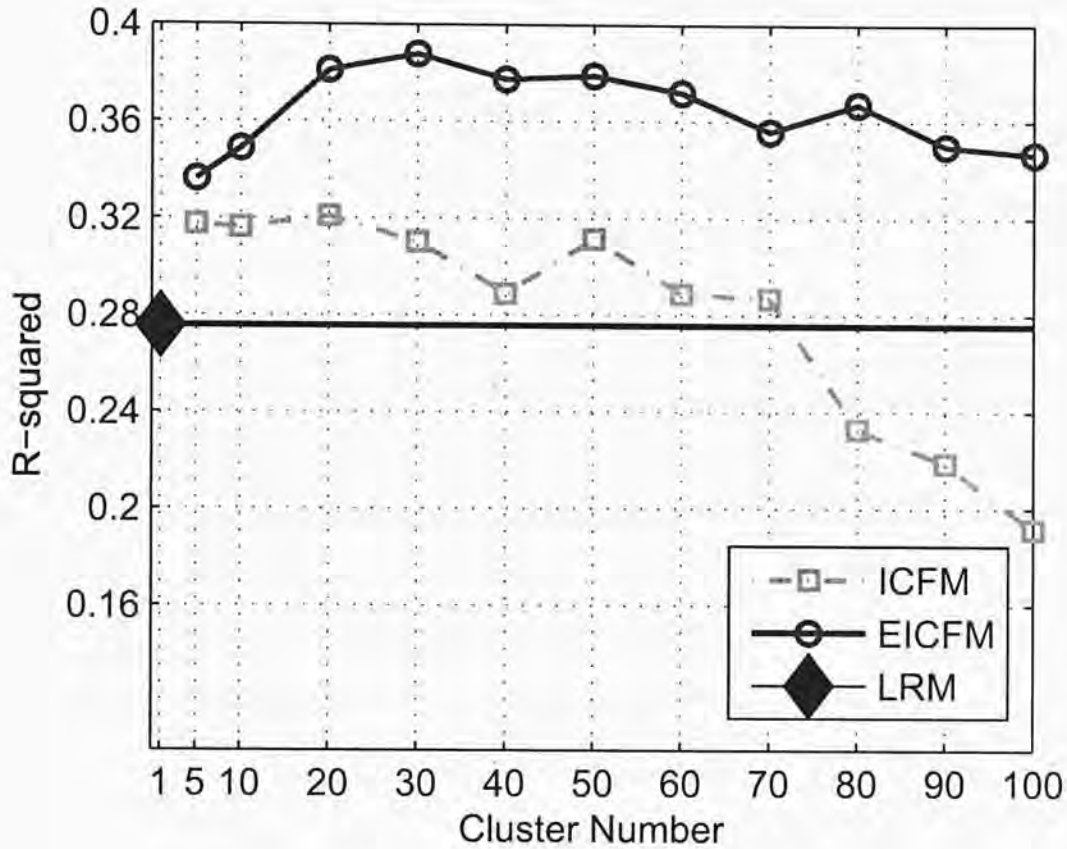


Figure 5.1: R^2 for user satisfaction prediction, in relation to the number of clusters M for the three prediction models. EICFM shows distinct improvement and is less sensitive to M .

number M for the three prediction models. Since LRM is unrelated to the cluster number, we represent the LRM result with a single diamond at $M = 1$. We observe that ICFM outperforms LRM for most values of M , and EICFM has the best performance throughout. In particular, when $M = 30$, R^2 values for EICFM, ICFM and LRM are 0.39, 0.31 and 0.27 respectively.

The improved performance from our CF models may result from the fact that local information is used to predict the ratings, rather than information from entire database, which may introduce noise to the prediction. Compared with ICFM, EICFM is even better, which may be due to EICFM's having a better balance between the influences from user style and system quality on

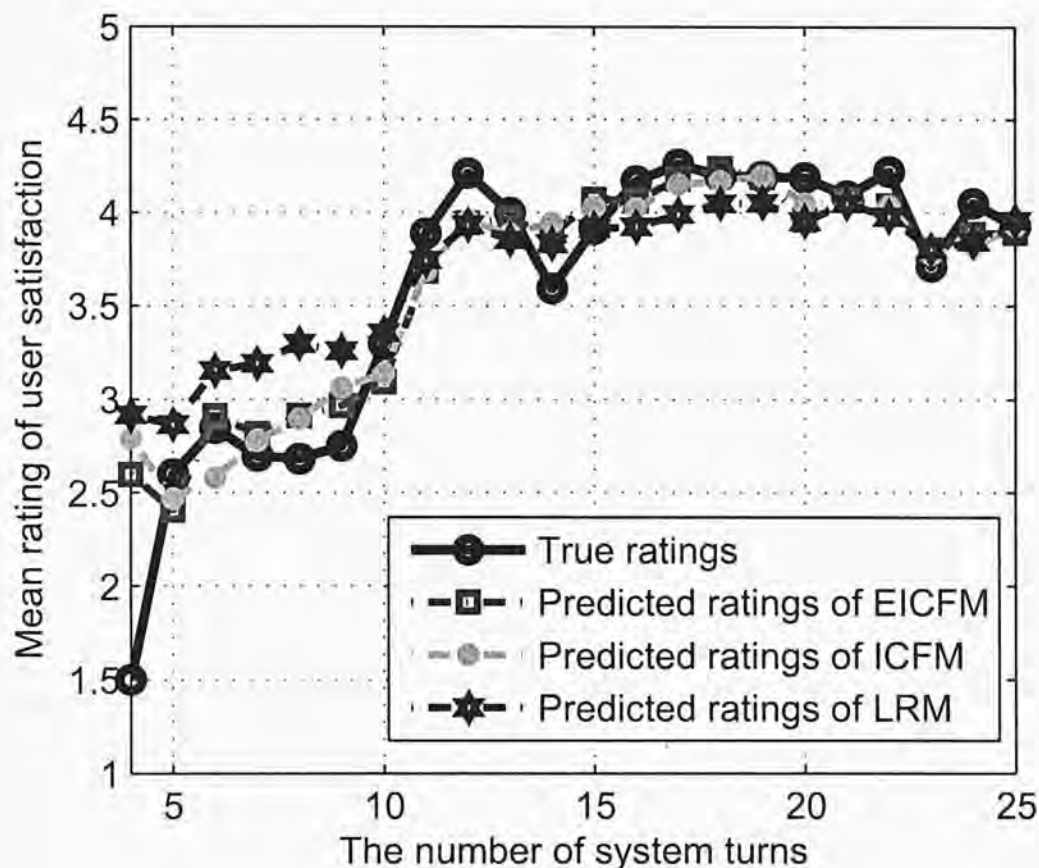


Figure 5.2: Average ratings of user satisfaction for dialogs over different #System Turns. The solid line with circle markers is for true ratings, and the other lines are for the predictions.

the overall judgment about the system. In addition, EICFM is more robust to the number of clusters than ICFM. The R^2 for ICFM drops below that of LRM when $M > 70$, while the performance of EICFM keeps stable within the same scale. This drop is reasonable. Because it requires certain number of samples to train a good regression model, the error increases as the cluster number becomes larger (hence samples in each cluster decrease). Further, ICFM drops quicker than EICFM due to its longer input feature vector, i.e., more samples are required in general.

In **Experiment II**, the three prediction models ($M = 30$ for both ICFM and EICFM) are trained on 4,000 dialogs, and are tested on the remaining

1,000 ones. We compare the average values of predicted and true ratings over **#System Turns**. In other words, the ratings are averaged over dialogs sharing the same **#System Turns**. This method compares ratings for groups of dialogs rather than single ones [26].

Figure 5.2 shows that both ICFM and EICFM can better reproduce the relation between ratings of user satisfaction and **#System Turns** than LRM. However, all the three models show a larger divergence between true ratings and predicted ones when **#System Turns** ≤ 4 . This divergence may be caused by the fact that there are fewer such training dialogs (around 10), which makes the prediction models do not fit well when **#System Turns** ≤ 4 .

Moreover, the plots of true ratings and predicted ones from ICFM and EICFM all show that the ratings of user satisfaction are at a low level (less than 3) and decrease when **#System Turns** < 10 . This is a reasonable result by considering the characteristics of the Let's Go! system. As Table 4.1 shows, the system has to get enough information from the user, such as the bus number, the origin, destination and departure time, in order to retrieve information from the database and provide corresponding results. After the user provides the requested information, the system also has to confirm each piece of information according to an explicit confirmation strategy. Hence, due to the design of the dialog model, the dialogs with fewer system turns (less than 10) prone to failure and get low ratings of user satisfaction.

5.4.2 Analysis of Prediction Results

To better understand the relations between user satisfaction and dialog metrics, we analyze the prediction results from EICFM. Based on the prediction ratings of 1,000 dialogs from EICFM in **Experiment II** in Section 5.4.1, we divide the evaluated dialogs into three categories: *A* (ratings in $[3.5, 5]$), *B* (ratings in $[2.5, 3.5)$) and *C* (ratings in $[1, 2.5)$). Fig. 5.3 shows the probability density

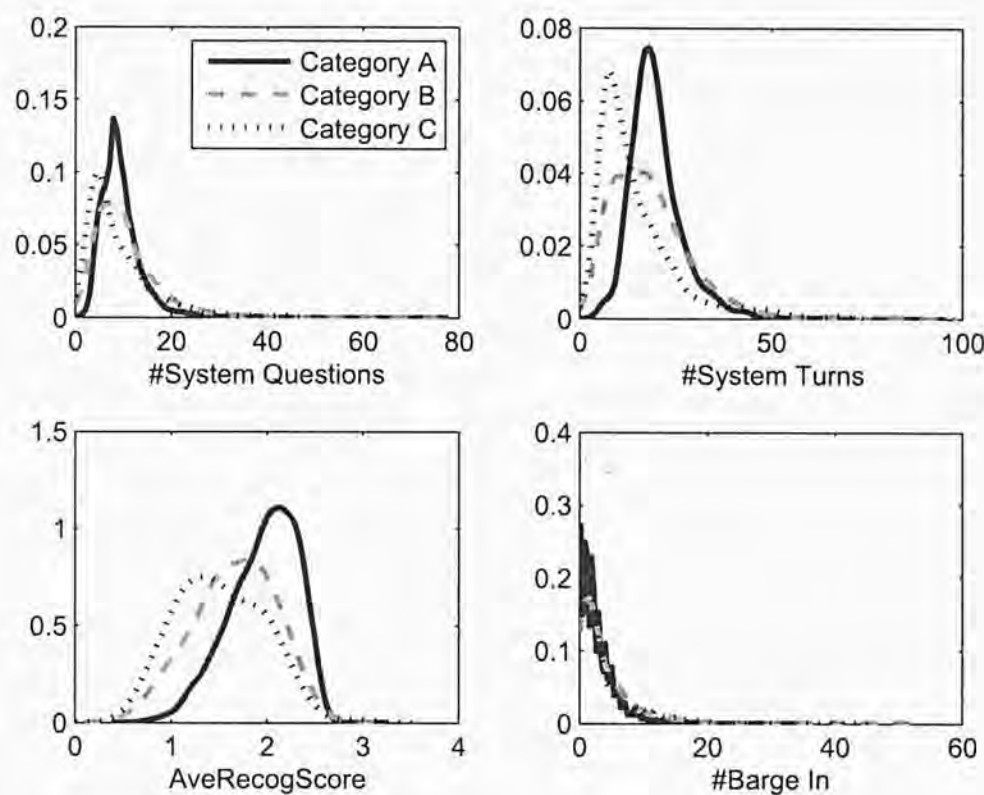


Figure 5.3: The probability density plots of **#System Questions**, **#System Turns**, **AveRecogScore** and **#Barge In** for dialogs rated high (*A*), medium (*B*) and low (*C*). The plots of other features are similar to that of **#Barge In**. The densities of *A*, *B* and *C* on AveRecogScore, #System Turns and #System Questions are different, which implies the three features have relatively larger impact on user satisfaction.

plots of 4 different features for dialogs in each category. We can see that the densities of A , B and C on AveRecogScore, #System Turns and #System Questions are different, which implies the three features have relatively larger impact on user satisfaction. Dialogs with appropriately higher values of these features tend to get higher ratings. The value of weight w in EICFM is 0.1, which indicates that system-related features are more helpful than user-related ones in determining user satisfaction.

In addition, we analyze the dialog discourse in each category and obtain some interesting observations in category A . We originally hypothesize that a dialog would be more efficient (i.e., a shorter dialog with task completed) and get higher rating of user satisfaction if the user were “aggressive” in trying to provide all related information at once, such as “I want to take bus 61C from airport to murray at 10p.m.”. However, in A category there is only a small portion of such dialogs. Most of the dialogs in category A show the users provide one piece of information at a time, and they are guided by the system step by step. Table 5.2 shows such an example which begins with bus information in user’s turn. This discourse is the most typical of category A . Our hypothesis may be invalid because there tends to be more recognition errors in longer utterances when the user includes more information in a single turn. These recognition errors reduce users’ overall impressions of the system quality.

5.5 Verifying the Generalibility of CF Model

In Section 5.4, we trained the CF model based on a training set of Let’s Go! dialogs and tested its prediction accuracy on the other set of Let’s Go! dialogs, i.e., the model is trained and tested on the same system. In this section, we further examine the *generalization ability* of the CF model across multiple systems (but in the same bus information domain).

S:	<i>Welcome to the CMU Let's Go bus information system</i>
S:	<i>What can I do for you?</i>
U:	<i>61B</i>
S:	<i>The 61B. Is this correct?</i>
U:	<i>YES</i>
S:	<i>Where do you wanna leave from?</i>
U:	<i>BRADDOCK AND TALBOT</i>
S:	<i>Leaving from BRADDOCK AND TALBOT?</i>
U:	<i>YES</i>
S:	<i>Where do you wanna go?</i>
U:	<i>SWISSVALE</i>
S:	<i>Going to SWISSVALE?</i>
U:	<i>YES</i>
S:	<i>When are you going to take that bus?</i>
U:	<i>NOW</i>
S:	<i>You want the next bus. Is this correct?</i>
U:	<i>YES</i>
S:	<i>Hold on. Let me check that for you.</i>
S:	<i>The next 61B leaves TALBOT AVENUE AT FIFTH BRADDOCK HOSPITAL at 10:26 p.m. and arrives at WASHINGTON at 10:43 p.m.</i>

Table 5.2: A dialog example with a discourse that is the most typical of category A (the highly rated dialogs). The users provide one piece of information at a time, and they are guided by the system step by step.

We restrict our training set to Let's Go! dialogs, and test the trained CF model on the four SDC dialog corpora in Section 4.4.2. Dialog examples from the four SDC systems are shown in Table 5.3-5.6. The 5,000 rated dialogs used in Section 5.4 serve as the training set to build our CF model (ICFM with $M = 30$). In Section 4.4.2, all the dialogs of the four SDC systems are published on MTurk and are rated by MTurk Workers. These SDC dialogs are used as the test set in this experiment.

The user satisfaction (S_u) of each dialog (both training and testing) is still obtained by averaging Workers' answers to Q3 and Q4. Each dialog is represented by a feature vector consisting of **TaskCompletion**, **#SystemTurns** (number of system turns), **#UserTurns** (number of user turns), **#SystemQuestions**

S:	<i>Welcome to the Let's Go bus information system</i>
S:	<i>What can I do for you?</i>
U:	<i>What's the next 61B from BRADDOCK to SWISSVALE?</i>
S:	<i>The 61B. Is this correct?</i>
U:	<i>YES</i>
S:	<i>Leaving from BRADDOCK?</i>
U:	<i>YES</i>
S:	<i>Going to SWISSVALE?</i>
U:	<i>YES</i>
S:	<i>You want the next bus. Is this correct?</i>
U:	<i>YES</i>
S:	<i>Hold on. Let me check that for you.</i>
S:	<i>The next 61B leaves TALBOT AVENUE AT FIFTH BRADDOCK HOSPITAL at 10:26 p.m. and arrives at WASHINGTON at 10:43 p.m.</i>

Table 5.3: An example dialog from system 1.

(number of system's questions), **#UserQuestions** (number of user's questions), **#Help** (number of user's help requests) and **WPUT** (average number of words per user turn). These features are shared by the Let's Go! system and the SDC systems, so they are not exactly the same as those in Section 5.4. Except for **TaskCompletion** which is taken from Workers' annotations of Q2, the remaining features are automatically extracted from log files. All the features use z-norm scores.

Figure 5.4 shows the mean scores of S_u of the four SDC systems predicted by the CF model and rated by MTurk Workers, respectively. We can observe that in addition to the Let's Go! system (system 1), the two kinds of mean ratings of the other systems are also quite close. Though trained on only one specific system, our CF model generalizes well to multiple systems within the bus information domain.

According to the evaluation results, system 3 has achieved the highest score among the four systems. As shown in Table 5.5, it works effectively and efficiently by asking for at most 3 pieces of information (e.g., origin, destination, or time) from the user, and then typically provides implicit confirmation. This

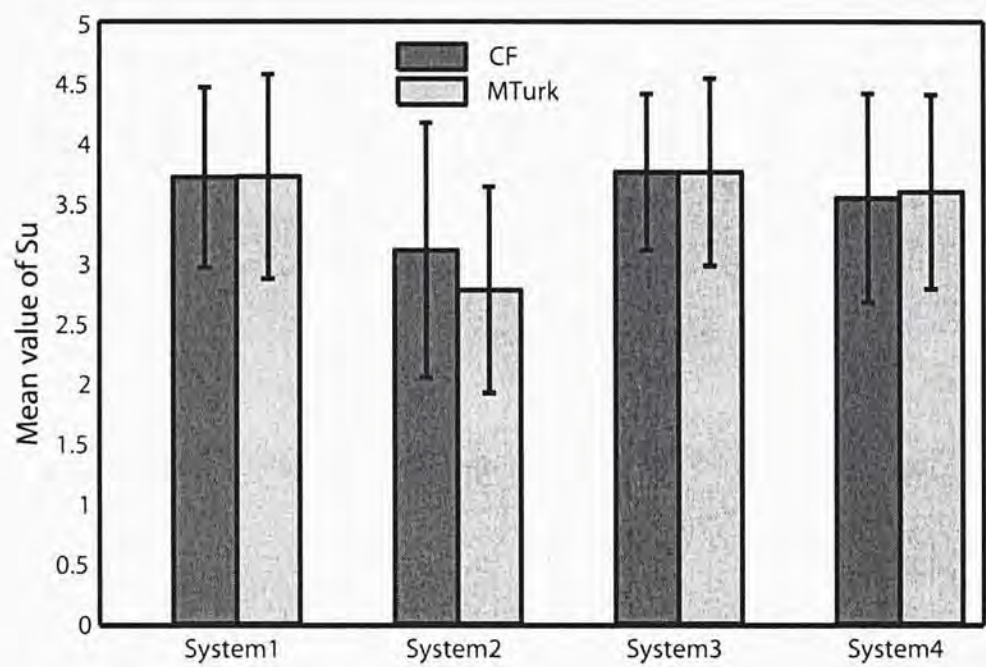


Figure 5.4: Evaluation results for the SDSs from CF and MTurk. The mean scores in terms of Su of the four SDC systems predicted by the CF model are quite similar to those rated by MTurk Workers

S:	<i>Welcome to the Pittsburgh bus information line.</i>
	<i>You can say start over anytime.</i>
	<i>Where are you going to take your bus from?</i>
U:	<i>um oh great are there any market</i>
S:	<i>did you mean "fifth avenue at market"?</i>
U:	<i>no</i>
S:	<i>Okay</i>
S:	<i>Where are you going to take your bus from?</i>
U:	<i>null</i>
S:	<i>I didn't quite catch that.</i>
S:	<i>Where are you going to take your bus from?</i>
U:	<i>null</i>
S:	<i>I didn't quite catch that.</i>
S:	<i>Where are you going to take your bus from?</i>
U:	<i>null</i>

Table 5.4: An example dialog from System 2.

S:	<i>Welcome to the Lets Go! bus information system for Pittsburgh.</i>
	<i>Where would you like to leave from?</i>
U:	<i>EAST OHIO STREET OPPOSITE CHESTNUT STREET</i>
S:	<i>Where do you wanna go?</i>
U:	<i>SOUTH SIDE EAST CARSON STREET TWENTY STREET</i>
S:	<i>When do you want to travel?</i>
U:	<i>TONIGHT AT NINE THIRTY</i>
S:	<i>There is a 54C leaving from east ohio street at chestnut at 9:33 p.m.</i>
	<i>It arrives at east carson street at twentieth in the south side at 10:15</i>
	<i>p.m. You may now say. next bus, previous bus, restart, or goodbye.</i>

Table 5.5: An example dialog from the system 3.

strategy leads to a high level of task completion and short dialogs. System 2 is the lowest scoring system whose dialogs usually cannot move forward in a smooth manner because of the bad recognition results, which can be observed in Table 5.4.

S:	<i>East Pittsburg Bus Schedules. Say a bus route, like 28X, or say I'm not sure.</i>
U:	<i>fifty four c</i>
S:	<i>Okay, 54C. To change, say go back. Where are you leaving from?</i>
U:	<i>bloomfield</i>
S:	<i>Okay, bloomfield. Where are you going to?</i>
U:	<i>corner of bausman and brownsville road</i>
S:	<i>I heard corner of bausman and brownsville road, but I don't have any routes that run from bloomfield to corner of bausman and brownsville. Say the neighborhood or place you're going to, or you can just hang up.</i>
U:	<i>mount oliver</i>
S:	<i>Okay, mount oliver. Do you want times for the next few buses?</i>
U:	<i>yeah</i>
S:	<i>To get the next few buses, say YES. Otherwise, say NO.</i>
U:	<i>yeah</i>
S:	<i>At 10:20 AM today there is a 54 C from PENN AVENUE and MAIN, in bloomfield, arriving BROWNSVILLE ROAD and BAUSMAN, in mount oliver, at 11:00 AM. Say repeat, next, previous, or start over.</i>

Table 5.6: An example dialog from the system 4.

5.6 Evaluation of The BusInfo System

In this section, we apply the CF model to evaluate the BusInfo system which is constructed based on POMDP in Chapter 3. This evaluation is conducted on the dialogs which are collected in Section 3.4.2.

We still train the CF model on the 5000 Let's Go! dialogs in Section 5.4, and take the average scores of Q3 and Q4 as the user satisfaction (S_u) of a dialog. Both Let's Go! (training) and BusInfo (testing) dialogs are represented with five dialog features of **TaskCompletion**, **#SystemTurns** (number of system turns), **#UserTurns** (number of user turns), **#SystemQuestions** (number of system's questions) and **WPUT** (average number of words per user turn).

After applying the trained CF model to the BusInfo dialogs (see Table 3.6) in Section 3.4.2, we get the mean score of user satisfaction on BusInfo of 4.65 (the standard deviation is 0.17). Figure 5.5 shows the normalized mean values

of dialog features for both Let's Go! and BusInfo systems. We can observe that generally BusInfo dialogs are shorter and have a higher task completion rate than Let's Go! system. The higher performance of BusInfo is mostly due to its simplicity which only allows users to input some simple queries on bus route, origin and destination. However, the higher task completion rate still provides the evidence that this POMDP-based system still shows its robustness to recognition uncertainties.

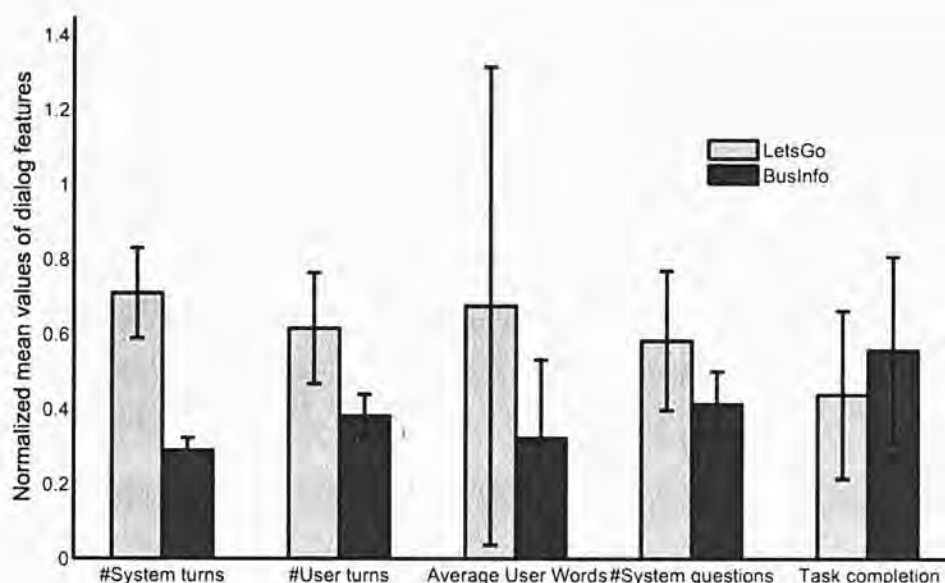


Figure 5.5: Normalized mean values with standard deviations of dialog features of Let's Go! and BusInfo systems.

5.7 Chapter Summary

In this chapter, we have presented an item-based collaborative filtering model (ICFM) for spoken dialog system evaluation, which is motivated by the idea of item-based recommendation that the rating of a dialog can be estimated from those of similar ones. In addition, ICFM is extended to EICFM by considering user judgments with respect of *user style* and *system quality*. These models

are applied to the dialog corpus from the Let's Go! system, the user judgments of which are collected with crowdsourcing on MTurk in Chapter 4.

Experimental results show both ICFM and EICFM can significantly improve the R^2 for prediction on test data when the cluster number M is set appropriately. In particular, R^2 for EICFM, ICFM and LRM are 0.39, 0.31 and 0.27 respectively when $M = 30$. Moreover, EICFM performs the best and is less sensitive to M than ICFM.

In addition, we have applied the CF model which is trained only on the Let's Go! dialog corpus to the SDC systems. The mean predicted user satisfaction scores of the four systems are quite close to those rated by MTurk Workers, which verifies the generalization ability of the CF evaluation model across multiple systems within the bus information domain. The CF model trained on the Let's Go! corpus is also used to evaluate the performance of the BusInfo system in Chapter 3, based on the dialogs collected in Section 3.4.2. Due to the high task completion rate and small number of dialog turns, the BusInfo receives a high score of user satisfaction from the CF model.

Chapter 6

Conclusions and Future Work

6.1 Thesis Summary

This thesis proposes an evaluation paradigm for spoken dialog system which consists of two components, *i.e.*, crowdsourcing user judgments on dialogs through Amazon Mechanical Turk (MTurk) and developing a collaborative filtering (CF) model for automatic evaluation based on the collected judgments. The summary of our work on the two perspectives will be presented in this section.

The ultimate objective of an SDS is to satisfy the demands of real users. Therefore, user's judgments are considered as the most important criterion for system performance. In this thesis, we present our initial attempt at using crowdsourcing to collect user judgments on spoken dialog systems through MTurk. Compared with the traditional method of user experiment which is expensive and time consuming, crowdsourcing has the advantages of cost-effectiveness, efficiency and flexibility. The comparison of MTurk Workers' and experts' annotations on SDSs shows a high level of agreement between the two groups, indicating that MTurk Workers have good faith in performing SDS evaluation and the crowdsourced data could be considered reliable.

Development of accurate evaluation models for automatically predicting the performance of SDSs is desirable for SDSs evaluation. In this thesis,

we propose an evaluation model, the CF model, to improve the accuracy of predicting user satisfaction of SDSs. It predicts user satisfaction of an unrated dialog by using the information of its rated neighbors. The basic CF model is extended by considering user judgments with respect of *system quality* and *user style*. Experimental results show that both basic and extended CF models can significantly improve the accuracy of predicting user satisfaction. In addition, we also verify the generalization ability of the CF model across multiple SDSs within the bus information domain.

The CF model is also applied to evaluate the performance of our bus information dialog system the BusInfo. The BusInfo system is built based on POMDP which is popular for statistical dialog modeling recently. BusInfo has a higher task completion rate and usually generates a moderate length of dialogs, since it is robust to uncertainties from speech recognition and trades off the effects of different system actions. Hence, it receives a high score of user satisfaction from our evaluation model.

6.2 Future Work

The work in this thesis can be further extended in several directions which are summarized as follows:

The crowdsourcing approach has a drawback that MTurk Workers are not the real users of the system. The Workers' quality perceptions of dialogs may not reflect the users' real needs and experiences with the system. Further research may consider providing access to SDSs through MTurk so that Workers can communicate with the system and evaluate their own interactions.

Moreover, the user judgments collected in this thesis are only limited to five aspects of SDS performance, *i.e.*, the user's confidence, the perceived task completion, the expected behavior, the overall performance and the categorization of task success. A thorough evaluation of an SDS needs to consider all dimensions

of the quality of an SDS. This work can be expanded in the future to cover as many dimensions as possible of the system quality, such as system intelligence or error recovery abilities.

In addition, the CF method proposed in this thesis cannot explicitly express the relations between user satisfaction and dialog metrics, although it improves the prediction accuracy of SDS performance. A possible extension to the current CF method is to utilize a unified model (*e.g.*, Bayesian network or MDP) to substitute for the linear models for the clusters. This extension will be useful to uncover the latent factors that explain the observed ratings of user satisfaction, and help us to gain insight into how different features influence the overall user satisfaction.

Bibliography

- [1] D. Bohus and A. Rudnicky, "RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda," in *Proc. of European Conference on Speech Communication and Technology*, 2003.
- [2] I. P. series Rec, "Parameters Describing the Interaction with Spoken Dialogue Systems," *ITU, Geneva*, 2005.
- [3] L. Hirschman, M. Bates, D. Dahl, W. Fisher, J. Garofolo, D. Pallett, K. Hunicke-Smith, P. Price, A. Rudnicky, and E. Tzoukermann, "Multi-site data collection and evaluation in spoken language understanding," in *Proc. of the workshop on Human Language Technology*, 1993, pp. 19–24.
- [4] A. Raux, B. Langner, D. Bohus, A. Black, and M. Eskenazi, "Let's go public! taking a spoken dialog system to the real world," in *Proc. of Interspeech*, 2005.
- [5] H. Meng, P. Ching, S. Chan, Y. Wong, and C. Chan, "ISIS: an adaptive, trilingual conversational system with interleaving interaction and delegation dialogs," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 11, no. 3, pp. 268–299, 2004.
- [6] Z. Wu, H. Meng, H. Ning, and S. Tse, "A corpus-based approach for cooperative response generation in a dialog system," *Chinese Spoken Language Processing*, pp. 614–626, 2006.

- [7] D. Litman, C. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembé, and S. Silliman, "Spoken versus typed human and computer dialogue tutoring," *International Journal of Artificial Intelligence in Education*, vol. 16, no. 2, pp. 145–170, 2006.
- [8] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [9] H. Meng, W. Lam, and C. Wai, "To believe is to understand," in *Proc. of European Conference on Speech Communication and Technology*, 1999.
- [10] H. Meng, C. Wai, and R. Pieraccini, "The use of belief networks for mixed-initiative dialog modeling," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 757–773, 2004.
- [11] W. Ward and S. Issar, "Recent improvements in the CMU spoken language understanding system," in *Proc. of the workshop on Human Language Technology*, 1994, pp. 213–216.
- [12] R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue *et al.*, *Survey of the state of the art in human language technology*. Citeseer, 1997.
- [13] T. Bateman, B. Kierstead, W. Noble, T. Curry, J. Lockett, L. Mersereau, and R. Ouellette, "System for coordinating communications via customer contact channel changing system using call centre for setting up the call between customer and an available help agent," Mar. 16 1999, uS Patent 5,884,032.
- [14] J. Mundt, "Interactive voice response systems in clinical research and treatment," *Psychiatric Services*, vol. 48, no. 5, p. 611, 1997.
- [15] R. Wolf, "Voice templates for interactive voice mail and voice response system," Jun. 23 1998, uS Patent 5,771,276.

- [16] F. Doshi and N. Roy, "Efficient model learning for dialog management," in *Proc. of the ACM/IEEE International Conference on Human-Robot Interaction*, 2007, pp. 65–72.
- [17] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proc. of Association for Computational Linguistics (ACL)*, 2000, pp. 93–100.
- [18] J. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.
- [19] B. Zhang, Q. Cai, J. Mao, E. Chang, and B. Guo, "Spoken dialogue management as planning and acting under uncertainty," in *Proc. of European Conference on Speech Communication and Technology*, 2001.
- [20] S. Möller, "Parameters for quantifying the interaction with spoken dialogue telephone services," in *Proc. of SIGdial Workshop on Discourse and Dialogue*, 2005, pp. 166–177.
- [21] M. McTear, "Spoken dialogue technology: enabling the conversational user interface," *ACM Computing Surveys (CSUR)*, vol. 34, no. 1, p. 169, 2002.
- [22] H. Bonneau-Maynard, L. Devillers, and S. Rosset, "Predictive performance of dialog systems," in *Proc. of Language Resources and Evaluation Conference (LREC)*, 2000.
- [23] S. Moller and J. Skowronek, "Quantifying the impact of system characteristics on perceived quality dimensions of a spoken dialogue service," in *Eighth European Conference on Speech Communication and Technology*, 2003.

- [24] M. Walker, D. Litman, C. Kamm, and A. Abella, "PARADISE: a framework for evaluating spoken dialogue agents," in *Proc. of Association for Computational Linguistics (ACL)*, 1997.
- [25] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella, "Evaluating spoken dialogue agents with PARADISE: Two case studies," *Computer Speech & Language*, vol. 12, no. 4, pp. 317–347, 1998.
- [26] K. Engelbrecht and S. Möller, "Pragmatic usage of linear regression models for the prediction of user judgments," in *Proc. of SIGdial Workshop on Discourse and Dialogue*, 2007.
- [27] K. Engelbrecht, F. Gödde, F. Hartard, H. Ketabdar, and S. Möller, "Modeling user satisfaction with Hidden Markov Model," in *Proc. of SIGDIAL Workshop on Discourse and Dialogue*, 2009, pp. 170–177.
- [28] I. McGraw, C. ying Lee, L. Hetherington, and J. Glass, "Collecting voices from the crowd," in *Proc. of Language Resources and Evaluation Conference (LREC)*, 2010.
- [29] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 254–263.
- [30] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: Automatic speech recognition with non-expert transcription," in *Proc. of Association for Computational Linguistics (ACL)*, 2010, pp. 207–215.
- [31] S. Xiaoyuan, M. Taghi *et al.*, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, 2009.

- [32] S. Seneff and J. Polifroni, "Dialogue management in the Mercury flight reservation system," in *ANLP/NAACL Workshop on Conversational systems-Volume 3*, 2000, pp. 11–16.
- [33] B. Zeigler and B. Bazor, "Dialog design for a speech-interactive automation system," in *Proc. of IEEE Workshop on Interactive Voice Technology for Telecommunications Applications*, 2002, pp. 113–116.
- [34] M. McTear, "Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit," *development*, vol. 5, p. 7, 1998.
- [35] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, "A form-based dialogue manager for spoken language applications," in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, vol. 2, 2002, pp. 701–704.
- [36] W. Xu and A. Rudnicky, "Task-based dialog management using an agenda," in *ANLP/NAACL 2000 Workshop on Conversational systems-Volume 3*, 2000, pp. 42–47.
- [37] S. Larsson and D. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, pp. 323–340, 2000.
- [38] J. Bos, E. Klein, O. Lemon, and T. Oka, "DIPPER: Description and formalisation of an information-state update dialogue system architecture," in *Proc. of SIGdial Workshop on Discourse and Dialogue*, 2003, pp. 115–124.
- [39] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2002.

- [40] M. Walker, J. Fromer, and S. Narayanan, "Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email," in *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, 1998, pp. 1345–1351.
- [41] N. Roy, J. Pineau, and S. Thrun, "Spoken dialog management for robots," in *The Proc. of Association for Computational Linguistics (ACL)*, 2000.
- [42] J. Williams and S. Young, "Scaling POMDPs for spoken dialog management," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2116–2129, 2007.
- [43] T. Bui, M. Poel, A. Nijholt, and J. Zwiers, "A tractable DDN-POMDP approach to affective dialogue modeling for general probabilistic frame-based dialogue systems," *Knowledge and Reasoning in Practical Dialogue Systems*, p. 34, 2009.
- [44] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.
- [45] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech & Language*, vol. 24, no. 4, pp. 562–588, 2010.
- [46] K. Hone and R. Graham, "Subjective assessment of speech-system interface usability," in *Seventh European Conference on Speech Communication and Technology*, 2001.
- [47] I. Rec, "P. 851. Subjective Quality Evaluation of Telephone Services Based on Spoken Dialogue Systems," *International Telecommunication Union, Geneva*, 2003.

- [48] J. Glass, J. Polifroni, S. Seneff, and V. Zue, "Data collection and performance evaluation of spoken dialogue systems: the MIT experience," in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, 2000.
- [49] P. Cohen, *Empirical methods for artificial intelligence*. MIT press, 1995, vol. 55.
- [50] K. Forbes-Riley and D. Litman, "Modelling user satisfaction and student learning in a spoken dialogue tutoring system with generic, tutoring, and user affect parameters," in *Proc. of Association of Computational Linguistics (ACL)*, 2006, pp. 264–271.
- [51] M. Walker, R. Passonneau, and J. Boland, "Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems," in *Proc. of Association for Computational Linguistics (ACL)*, 2001, p. 522.
- [52] C. Kamm, M. Walker, and D. Litman, "Evaluating spoken language systems," in *Proc. of AVIOS*, 1999.
- [53] M. Walker, C. Kamm, and D. Litman, "Towards developing general models of usability with paradise," *Natural Language Engineering*, vol. 6, no. 3&4, pp. 363–377, 2000.
- [54] N. Beringer, U. Kartal, K. Louka, F. Schiel, U. Türk *et al.*, "Promise: A procedure for multimodal interactive system evaluation," in *Proceedings of the Workshop Multimodal Resources and Multimodal Systems Evaluation*, 2002, pp. 90–95.
- [55] L. Dybkjaer, N. Bernsen, and W. Minker, "Evaluation and usability of multimodal spoken language dialogue systems," *Speech Communication*, vol. 43, no. 1-2, pp. 33–54, 2004.

- [56] J. Williams, P. Poupart, and S. Young, "Factored partially observable Markov decision processes for dialogue management," in *Proc. of Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.
- [57] O. Pietquin, "A framework for unsupervised learning of dialog strategies," in *Faculty of Engineering, Mons(TCTS Lab)*, 2004.
- [58] Z. Yang, B. Li, Y. Zhu, I. King, G. Levow, and H. Meng, "Collection of User Judgments on Spoken Dialog System with Crowdsourcing," in *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*, 2010.
- [59] M. Shoukri, *Measures of interobserver agreement*. CRC Press, 2004.
- [60] J. Landis and G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.
- [61] A. Black, S. Burger, B. Langner, G. Parent, and M. Eskenazi, "Spoken dialog challenge 2010," in *Proc. of IEEE Workshop on Spoken Language Technology*, 2010.
- [62] Z. Yang, Y. Zhu, B. Li, I. King, G. Levow, and H. Meng, "Collaborative Filtering Model for User Satisfaction Prediction in Spoken Dialog Evaluation," in *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*, 2010.
- [63] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on Knowledge and Data Engineering*, pp. 734–749, 2005.
- [64] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. of International Conference on World Wide Web (WWW)*, 2001, p. 295.

CUHK Libraries



004865827